Computational methods for simulating and parameterizing nucleic acid secondary structure thermodynamics and kinetics

Thesis by
Mark Evan Fornace

In Partial Fulfillment of the Requirements for the
Degree of
Doctor of Philosophy

CALIFORNIA INSTITUTE OF TECHNOLOGY
Pasadena, California

2022
Defended December 6, 2021

Mark Evan Fornace
ORCID: 0000-0002-5829-5839

ACKNOWLEDGEMENTS

I would first like to thank my committee. My advisor, Niles Pierce, has always led by example in showing the need for the hard and careful work which is required to make a novel idea into a truly rigorous and useful method. I would like to thank my committee chair, Tom Miller, for his continued support and tireless focus on getting to our shared goals. I would like to thank Erik Winfree and Garnet Chan for their supportive and fun discussions, as well as for their positivity towards even my most speculative ideas. Finally, I would like to thank Zhen-Gang Wang for the openminded viewpoints that he has offered.

I have been fortunate to work with many other excellent young scientists at Caltech. Much of the work in this thesis was done with the help of Marta Gonzalvo-Ulla, who has never given up in trying out our new methodologies, be they good or bad, and Nick Porubsky, who was an enjoyable officemate and frequent collaborator on algorithms and NUPACK. I would like to thank Jining Huang for his positivity, networking expertise, and custom-built clusters. Sherry (Lixue) Cheng was a close collaborator in our initial formulations of ideas for Gaussian process learning and regression. I will always appreciate Sam Schulte for his hard work and good humor in our multiplexing experiments and trials of both my useful and useless algorithms. I am grateful to Cody Newman for her software acumen and to Maayan Schwarzkopf and Harry Choi for their expertise in image acquisition and analysis.

I would like to thank Kaleigh Durst, Grace Shin, Zhewei Chen, Heyun Li, Mikhail Hanewich-Hollatz, Lisa Hochrein, and Melinda Kirk for fun and helpful conversations. Bergthor Traustason, my best and only SURF undergraduate intern, impressed me with his diligence and hard work throughout an exciting summer. I have enjoyed working with Duo Tao and Avinash Nanjundiah in their rotations and first years.

From Caltech, I am grateful for my time collaborating and talking with Joonho Lee and Eric Sundstrom, who taught me much of what I needed to know about electronic structure methods. From the Miller lab, Connie, Mike, and Michiel taught me the programming and chemical reasoning that I needed to transition into more and more theoretical work. I would also like to thank George Rossman for supporting me in changing departments and charting my own research path. And I would like to thank my undergraduate advisor Nicolas Dauphas for setting me off on my research career.

I am grateful to my friends Sebastian, Howard, Daichi, Jeff, Kristen, and Andrew for teaching me how to keep having fun during a long PhD. Last, but most importantly to me, I would like to thank my family for their support throughout graduate school, and Vicky and Willow for sticking around through it all.

ABSTRACT

Nucleic acid secondary structure models offer a simplified but powerful lens through which to view, analyze, and design nucleic acid chemistry. Computational approaches based on such models are central to current research directions across molecular programming, synthetic biology, and the life sciences more broadly. In this work, we develop a unified framework for constructing and understanding dynamic programming algorithms for complexes of interacting nucleic acid strands. Our framework combines three ingredients. First, we develop new recursions to include contributions from coaxial and dangle stacking in an efficient and principled way. Second, we formulate the concept of an evaluation algebra, which defines the mathematical form of each subproblem in the dynamic program. Whereas previous modeling efforts have relied on case-by-case handling of different thermodynamic quantities, we use evaluation algebras to elegantly and efficiently compute a variety of physical quantities using the same recursions. Third, we develop efficient operation orders for a variety of physical quantities of experimental interest. Combining our advances, we are able to achieve speedups of 20–120× and scalable calculations of complexes of up to 30,000 nucleotides. Our achievements promise to dramatically expand the scope and utility of computational analysis and design of nucleic acid thermodynamics.

While current dynamic programming algorithms achieve efficient computation of thermodynamic quantities for a given nucleic acid sequence, they do not provide kinetic information. Therefore, investigations of secondary structure kinetics rely on stochastic simulations of trajectories in secondary structure space. We improve upon these simulation methodologies to achieve lower computational complexities and large empirical speedups. We extend our algorithms to an ensemble which fully includes coaxial and dangle stacking states, expanding the scope of the kinetic analysis that is currently possible.

Current secondary structure models are parametrized using thermodynamic information gleaned from decades of melt experiments of RNA and DNA in specific experimental conditions. Only rough kinetic information is currently available from past experiments, and information on solvent and material dependence is lacking. We develop a fully computational approach based on Gaussian processes and molecular dynamics in order to provide a generic method for estimating thermodynamic and kinetic parameters, applicable conceptually to any nucleic acid material and experimental setting of interest. Our methodology offers an atomistic view of nucleic acid base pairing and faithfully reproduces most experimental data. It thus provides a powerful black-box approach for extensibly calculating the kinetic and thermodynamic parameters that secondary structure models require.

PUBLISHED CONTENT AND CONTRIBUTIONS

Mark E. Fornace, Nicholas J. Porubsky, and Niles A. Pierce. A unified dynamic programming framework for the analysis of interacting nucleic acid strands: Enhanced models, scalability, and speed. *ACS Synthetic Biology*, 9(10):2665– 2678, 2020. doi: 10.1021/acssynbio.9b00523.
URL `https://doi.org/10.1021/acssynbio.9b00523`. PMID: 32910644.

MEF developed and implemented the evaluation algebra framework, blockwise operation order, and the backtrack-free pair probability algorithm. MEF implemented the dynamic programming algorithms, including implementation of all recursions without coaxial and dangle stacking. NJP developed the additional recursions necessary to capture coaxial and dangle stacking states. MEF and NJP implemented those recursions. MEF and NJP developed and implemented the simultaneous sampling algorithm and suboptimal structure generation algorithms. MEF conducted the benchmark calculations, with NJP generating the designed sequences used in selected benchmarks. MEF, NJP, and NAP drafted, reviewed, and edited the text.

CONTENTS

Acknowledgements . . . iii
Abstract . . . iv
Published Content and Contributions . . . v
Contents . . . v
List of Figures . . . ix
List of Tables . . . xii
Chapter I: Introduction . . . 1
1.1 Nucleic acid secondary structures . . . 2
1.1.1 Unpseudoknotted structures . . . 3
1.2 Dynamic programming algorithms . . . 4
1.2.1 Abstract evaluation algebras . . . 5
1.3 Simulation of secondary structure kinetics . . . 6
1.3.1 Stochastic simulation . . . 7
1.3.2 Kinetics including coaxial and dangle stacking contributions . . . 7
1.4 Computational parametrization of secondary structure models . . . 8
1.4.1 Simulation design and choice . . . 9
1.4.2 Molecular dynamics methodologies . . . 9
1.4.3 Thermodynamic and kinetic regression . . . 10
1.5 Conclusions . . . 10
Bibliography . . . 11
Chapter II: Improved algorithms for the equilibrium analysis of nucleic acid complexes . . . 13
2.1 Physical model . . . 16
2.1.1 Complex ensemble and test tube ensembles . . . 16
2.1.2 Loop-based free energy model . . . 17
2.1.3 Coaxial and dangle stacking subensembles within complex ensembles 19
2.1.4 Symmetry correction . . . 20
2.1.5 Free energy parameters . . . 20
2.2 Algorithm . . . 21
2.2.1 Physical quantities . . . 21
2.2.2 Existing dynamic programs . . . 26
2.2.3 Unified dynamic programming framework . . . 27
2.2.4 Recursions for the complex ensemble with coaxial and dangle stacking 28
2.2.5 Evaluation algebras for partition function, minimum free energy, and ensemble size . . . 29
2.2.6 Overflow-safe evaluation algebra for large partition function calculations . . . 30
2.2.7 Efficient blockwise dynamic programs over subcomplexes using caching and vectorization . . . 32

2.2.8 Enhanced efficiency and scalability of the partition function algorithm for complex ensembles including very large complexes 34
2.2.9 Enhanced efficiency of the partition function algorithm for sets of complexes in test tube ensembles 34
2.2.10 Backtrack-free base-pairing probability matrices 35
2.2.11 Evaluation algebras and backtracking operation orders for simultaneous structure sampling, MFE structure determination, and suboptimal structure determination 38
2.3 Conclusions 40
2.4 Methods summary 41
2.4.1 Implementation 41
2.4.2 Trials 41
2.5 Resources 41
2.5.1 NUPACK source code 41
2.5.2 NUPACK Python module 41
Bibliography 42
Chapter III: Efficient simulation of nucleic acid secondary structure stochastic kinetic trajectories 45
Chapter IV: Molecular dynamics methods for simulating nucleic acid base-pairing reactions 46
Chapter V: Computational parameterization of equilibrium and kinetic nucleic acid secondary structure models 47
Appendix A: Additional details for improved algorithms for the equilibrium analysis of nucleic acid complexes 48
S1 Additional free energy model details 48
S1.1 Strand association penalty for a complex 48
S1.2 Salt corrections for DNA complexes 48
S1.3 Temperature dependence 50
S1.4 Treatment of constant free energy terms for complex ensembles . . 50
S1.5 RNA and DNA parameter sets 51
S1.6 Historical RNA and DNA parameter sets (for backwards compatibility with NUPACK 3) 52
S1.7 Functional form of RNA and DNA free energy models 54
S2 Recursions for the complex ensemble with or without coaxial and dangle stacking 61
S2.1 Separate recursions for intrastrand and interstrand blocks 61
S2.2 Conventions for recursion diagrams and equations 62
S2.3 Recursions without coaxial and dangle stacking subensembles . . . 67
S2.4 Recursions for interior loop contributions 83
S2.5 Approximate dangle stacking without coaxial stacking (for backwards compatibility with NUPACK 3) 90
S2.6 Recursions with coaxial and dangle stacking subensembles 90
S3 Evaluation algebras for each physical quantity 126
S3.1 Evaluation algebras for scalar outputs 127
S3.2 Evaluation algebras for structure generation 131

S4 Operation orders for each physical quantity . . . 137
S4.1 A partial order on recursion elements . . . 137
S4.2 Operation order for partition function, structure count, and MFE . . 138
S4.3 Operation order for equilibrium pair probability matrix . . . 142
S4.4 Operation order for sampled structure generation . . . 144
S4.5 Operation order for interior loops in backtracking algorithms . . . 147
S4.6 Operation order for suboptimal structure generation . . . 149
S5 Distinguishability issues . . . 151
S5.1 Partition function . . . 151
S5.2 Equilibrium secondary structure probability . . . 154
S5.3 Equilibrium base-pairing probabilities . . . 155
S5.4 Structure sampling . . . 156
S5.5 Equilibrium complex concentrations . . . 157
S5.6 Ensemble pair fractions . . . 157
S5.7 MFE free energy and secondary structure . . . 159
S6 Additional studies . . . 162
S6.1 Comparison of predictions to structure databases . . . 162
S6.2 Empirical dependence of ensemble size on complex size . . . 164
S6.3 Empirical dependence of partition function on complex size . . . 166
S6.4 Relative cost of partition function, equilibrium pair probability, and MFE calculations . . . 169
S6.5 Speed and scalability of partition function calculations with different floating point formats and evaluation algebras . . . 170
S6.6 Performance of simultaneous vs sequential structure sampling . . . 171
S6.7 Comparison of deterministic vs random MFE proxy structure estimation . . . 179
S7 Validation . . . 181
S7.1 Unit tests . . . 181
S7.2 Regression tests . . . 183
S7.3 Exhaustive enumeration algorithms . . . 185
Bibliography . . . 192

LIST OF FIGURES

Number		*Page*
1.1	Primary, secondary, and tertiary structures of nucleic acids	1
1.2	Secondary structure notation	3
1.3	Evaluation algebra conceptualization	5
1.4	Example base pairing reactions simulated using molecular dynamics	8
2.1	Complex and test tube ensembles	15
2.2	Loop-based free energy model for a complex	18
2.3	Coaxial and dangle stacking states for multiloops and exterior loops	18
2.4	Operation order for partition function dynamic program over a complex ensemble with N nucleotides	22
2.5	Partition function dynamic program recursion diagrams and recursion equations	23
2.6	Unified dynamic programming framework	27
2.8	Blockwise operation order for dynamic programs operating on complex and test tube ensembles	32
2.9	Conceptual interplay between three dynamic program ingredients	33
2.10	Enhanced efficiency for partition function calculations on complex ensembles including very large complexes	34
2.11	Overflow-safe partition function calculations	35
2.12	Enhanced efficiency of the partition function algorithm for sets of complexes in test tube ensembles	36
2.13	Equilibrium test tube analysis in under 1 minute	37
2.14	Backtrack-free calculation of the equilibrium base-pairing probability matrix	38
2.15	Enhanced efficiency for sampling multiple structures through simultaneous sampling	39
S1	Separate recursions for intrastrand and interstrand blocks	62
S2	Nomenclature and connectivity for recursions without coaxial and dangle stacking	68
S3	$R^{\varnothing}_{\text{INTRA}}$ recursion without coaxial and dangle stacking	69
S4	R^{s}_{INTRA} recursion without coaxial and dangle stacking	71
S5	R^{b}_{INTRA} recursion without coaxial and dangle stacking	72
S6	R^{ms}_{INTRA} recursion without coaxial and dangle stacking	73
S7	R^{m}_{INTRA} recursion without coaxial and dangle stacking	74

S8 $R^{\varnothing}_{\text{INTER}}$ recursion without coaxial and dangle stacking 76
S9 R^{s}_{INTER} recursion without coaxial and dangle stacking 78
S10 R^{b}_{INTER} recursion without coaxial and dangle stacking 79
S11 R^{ms}_{INTER} recursion without coaxial and dangle stacking 81
S12 R^{m}_{INTER} recursion without coaxial and dangle stacking 82
S13 Nomenclature and connectivity for recursions with coaxial and dangle stacking 92
S14 Summation over individual dangle states 93
S15 $R^{\varnothing}_{\text{INTRA}}$ recursion with coaxial and dangle stacking 94
S16 R^{s}_{INTRA} recursion with coaxial and dangle stacking 96
S17 R^{cd}_{INTRA} recursion with coaxial and dangle stacking 97
S18 R^{b}_{INTRA} recursion with coaxial and dangle stacking 99
S19 R^{ms}_{INTRA} recursion with coaxial and dangle stacking 103
S20 R^{mcs}_{INTRA} recursion with coaxial and dangle stacking 104
S21 R^{mc}_{INTRA} recursion with coaxial and dangle stacking 105
S22 R^{md}_{INTRA} recursion with coaxial and dangle stacking 106
S23 R^{m}_{INTRA} recursion with coaxial and dangle stacking 107
S24 $R^{\varnothing}_{\text{INTER}}$ recursion with coaxial and dangle stacking 109
S25 R^{s}_{INTER} recursion with coaxial and dangle stacking 110
S26 R^{cd}_{INTER} recursion with coaxial and dangle stacking 111
S27 R^{b}_{INTER} recursion with coaxial and dangle stacking 113
S28 R^{n}_{INTER} recursion with coaxial and dangle stacking 120
S29 R^{ms}_{INTER} recursion with coaxial and dangle stacking 120
S30 R^{mcs}_{INTER} recursion with coaxial and dangle stacking 122
S31 R^{mc}_{INTER} recursion with coaxial and dangle stacking 123
S32 R^{md}_{INTER} recursion with coaxial and dangle stacking 124
S33 R^{m}_{INTER} recursion with coaxial and dangle stacking 124
S34 Blockwise operation order 137
S35 Illustration of simultaneous sampling operation order 144
S36 Example secondary structures and polymer graphs for a complex with strand ordering π =AAAA 153
S37 Empirical dependence of ensemble size on complex size 165
S38 Dependence of partition function on complex size for random and designed sequences 167
S39 Example MFE proxy structures for random and designed sequences 168
S40 Relative cost of partition function, equilibrium pair probability, and MFE calculations 169

S41 Speed and scalability of partition function calculations with different floating point formats and evaluation algebras 170
S42 Cost of simultaneous and sequential structure sampling for random complexes for different numbers of samples 172
S43 Sampling cost as a function of complex size (N) for random complexes . . . 173
S44 Sampling cost as a function of number of samples (J) for random complexes 173
S45 Cost of simultaneous and sequential structure sampling for designed complexes for different numbers of samples 174
S46 Sampling cost as a function of complex size (N) for designed complexes . . . 175
S47 Sampling cost as a function of number of samples (J) for designed complexes 175
S48 Comparison of deterministic vs random MFE proxy structure estimation . . . 180

LIST OF TABLES

Number *Page*

2.1 Algorithmic ingredients for calculating diverse physical quantities. 25
2.2 Evaluation algebras for dynamic programming algorithms 29
A.1 Regression of multiloop parameters for `rna06` 59
A.2 Comparison of predictions to structure databases 164
A.3 Bivariate least-squares linear regression of sampling complexity 176
A.4 Univariate least-squares linear regression of sampling complexity in J 177
A.5 Univariate least-squares linear regression of sampling complexity in N . . . 178

Chapter 1

INTRODUCTION

DNA and RNA are co-polymeric molecules consisting of sequences of nucleotides linked by base pairs consisting of multiple hydrogen bonds. Biological RNA consists of adenine (A), cytosine (C), guanine (G), and uracil (U) bases. Uracil is replaced by thymine (T) in DNA. States of DNA and RNA species are amenable to description by three main levels of classification (Figure 1.1):

1. *Primary structure*: the sequences of bases {A, C, U, G, T} defining each covalently bonded strand in the state, occasionally extended to incorporate chemical derivatives (e.g. by methylation).

2. *Secondary structure*: given a primary structure, the set of hydrogen bonding base pairs in the state, usually either Watson-Crick base pairs (A·U, A·T, C·G) or, less frequently, "wobble" pairs (G·U).

3. *Tertiary structure*: given primary and secondary structures, the geometric configuration of nucleic acid strands in a state including helicity, atomic positions, and other bonds.

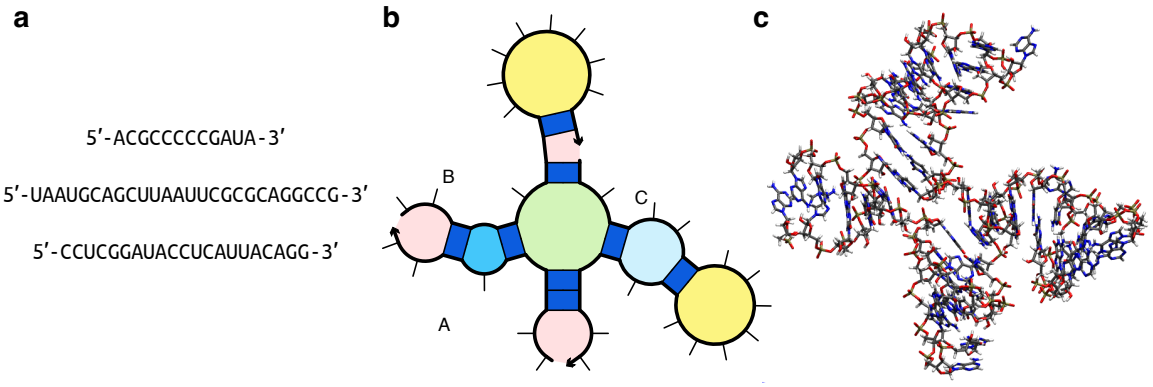


Figure 1.1: Primary, secondary, and tertiary structures of nucleic acids. (a) Example of primary structure denoted by 3 RNA strands (A, B, C) listed by their bases in 5′ to 3′ order. (b) A corresponding secondary structure, showing the phosphate backbone (thick lines), unpaired bases (ticks), and base pairs (thin lines). (c) A corresponding tertiary structure, colored by atom identity.

Compared to other biochemical materials, including proteins, nucleic acids are somewhat easier to describe solely at the primary or secondary structure level [3]. While protein thermodynamics require tertiary fold information, the strong affinity and specificity of nucleic acid base hydrogen bonding means that nucleic acid thermodynamics can often be analyzed effectively by base pairing information alone. Genomic and informatic analyses commonly rely solely on primary structures, i.e. raw sequence information. On the other hand, bioengineering approaches rely more commonly on secondary structure, with primary structure a constant on experimental timescales. Experimentalists commonly seek to analyze the ensemble of predicted secondary structures given a primary structure (*sequence analysis*) or to optimize a primary structure given a desired ensemble of secondary structures (*sequence design*). The effectiveness of current algorithms enables *molecular programming*, in which computer science principles and abstractions are applied to the design of nucleic acid systems that accomplish reaction pathway engineering [19], complex molecular structure folding [13], biological circuit design [12], and other objectives.

1.1 Nucleic acid secondary structures

After the discoveries of the molecular structure and importance of RNA and DNA, it was soon realized that secondary structures could be defined in terms of neighboring loops [8]. It was furthermore established that a rough but useful model of secondary structure energetics might be derived based on a nearest-neighbor loop-based model [17]. Such a model can encompass the most important energetics of nucleic acid base pairing, including (most importantly) terms for hydrogen bonding between complementary bases in a base pair and π-π stacking of adjacent bases. In such a model, the free energy of a secondary structure is parametrized as the sum of loop free energies, which in turn are defined solely by the identities and order of the bases within the loop, i.e.

$$\Delta G(\phi, s) = \sum_{\text{loop} \in s} \Delta G(\text{loop}) + \text{const} \tag{1.1}$$

where s is a given secondary structure for sequence ϕ. In Figure 1.1b, an example secondary structure is decomposed into loops depicted as contiguous colored regions. In traditional nearest-neighbor free energy models, a loop is grouped according to its number of bounding base pairs and whether or not it contains a strand break. For both physical and algorithmic reasons, different functional forms of $\Delta G(\text{loop})$ are used for loops containing a strand break (*exterior loops*), loops containing a single base pair (*hairpin loops*), loops containing two base pairs (*interior loops*), and loops containing at least three base pairs (*multiloops*) [4, 7].

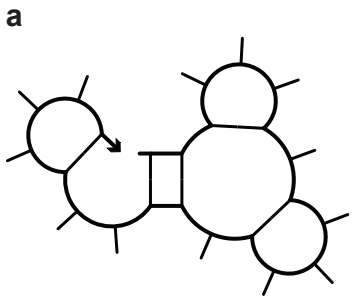


Figure 1.2: Secondary structure notation. (a) Example (single-stranded) secondary structure. (b) Equivalent dot-parens structure notation. Proceeding along the backbone in a clockwise direction in (a), the opening of a base pair is denoted by "(", the closing of a base pair by ")", and an unpaired base by ".".

1.1.1 Unpseudoknotted structures

The information reduction of tertiary structure (Figure 1.1c) to secondary structure (Figure 1.1b) is vast. Yet it is still extremely difficult to calculate physical quantities of interest over a fully general ensemble of all possible secondary structures. The number of possible secondary structures for a system of N nucleotides grows combinatorially with respect to N. As counting the number of secondary structures is in essence the same as counting the number of matchings in a general graph, fully general summation over such an ensemble is #P-hard, one of the most inapproachably difficult classes of computational complexity [18].

Further simplification comes at the cost of ignoring certain secondary structures referred to as pseudoknotted [16]. For a given ordered sequence of nucleotides making up a primary structure, a pseudoknotted structure results when there are two base pairs indexed (i, j) and (k, l) such that $i < k < j < l$. Pseudoknots may be visualized as crossing base pairs on a secondary structure's polymer graph (e.g. [3]). Algorithmically, a lack of pseudoknots implies that there are no further interactions between two regions of a secondary structure which are separated by a single base pair.

Even excluding pseudoknots, the number of secondary structures compatible with a given ordering of strands grows exponentially in the number of nucleotides (see Figure S37). If one imagines, fictitiously, that any base may pair to any other base, then counting secondary structures is isomorphic to counting Motzkin paths, a type of lattice path in which the path may change by increments of -1, 0, or +1 and is bounded above 0 [11]. In this scenario, the number of secondary structures of N nucleotides is precisely given by the Motzkin number M_N [5], which is asymptotically proportional to $3^N N^{-3/2}$. The isomorphism may be seen by considering the height of the Motzkin path to be the number of open base pairs as one progresses from $5'$ to $3'$ in a dot-parens string (like that in Figure 1.2b).

1.2 Dynamic programming algorithms

Suppose we are interested in the free energy of a given complex of nucleic acid strands. The *complex partition function* measures the stability of a given complex and may be used to solve for equilibrium complex concentrations in a test tube [4]. By the basic statistical mechanics of the isothermal-isobaric ensemble, the complex partition function may be calculated as a Boltzmann summation over the ensemble $\Gamma(\phi)$ of secondary structures compatible with sequence ϕ:

$$Q(\phi) = \sum_{s \in \Gamma(\phi)} \exp(-\Delta G(\phi, s)/k_B T) \tag{1.2}$$

where k_B is the Boltzmann constant, T the temperature, and $\Delta G(\phi, s)$ the structure free energy. Direct enumeration of all secondary structures incurs $e^{O(N)}$ computational complexity in the number of nucleotides N. Such an approach is infeasible for all but the smallest complexes of interest. A superior solution is to perform this summation using a dynamic programming algorithm, using the partition function of each subsequence of the complex to solve for progressively larger subsequences and culminating in the partition function of the complete complex. A dynamic programming algorithm builds from solutions of the smallest subproblems to the complete problem.

To illustrate the concept, consider the dynamic programming algorithm for a simplified nucleic acid free energy model in which each the free energy is simply a sum over flat contributions from each type of base pair. This algorithm might be defined as working via the following single recursion:

$$Q_{i,j} = \begin{cases} Q_{i,j-1} + \sum_{k=i}^{j-1} Q_{i,k-1} Q_{k+1,j-1} \exp(-\Delta G_{k,j}/k_B T) & i < j \\ 1 & \text{otherwise.} \end{cases} \tag{1.3}$$

Here, $Q_{i,j}$ is the partition function of the subsequence $[i{:}j]$, and $\Delta G_{i,j}$ is the free energy from base pairing bases i and j. The idea is that in increasing the size of the considered subsequence by one (by including base j), the possible configurations are those of $[i{:}j\text{-}1]$ plus configurations in which j is paired to a base in $[i{:}j\text{-}1]$. Each of the contributions where j is paired to k may be decomposed into a product of partition functions on either side of the $k \cdot j$ base pair. More complicated (and accurate) free energy models for secondary structure add many more types of energetic contributions, including stacking energies and other terms [7].

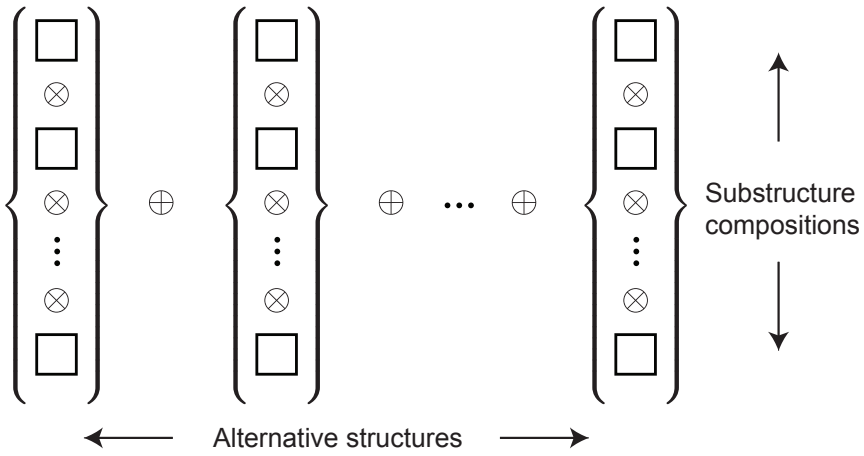


Figure 1.3: Evaluation algebra conceptualization. The *evaluation algebra* captures the generic idea of summation over alternative secondary structures to compute a single physical quantity of interest for a given set of strands. In the nearest-neighbor model, the energetics of different loops within a given secondary structure are assumed independent of each other, which, for partition function calculation, leads to the distribution of multiplication over addition in the semiring.

1.2.1 Abstract evaluation algebras

Calculation of complex free partition functions is perhaps one of the most essential tools for understanding nucleic acid thermodynamics. Knowledge of these free energies, for instance, can be used to solve for the equilibrium complex concentrations in a test tube [4]. However, there are many more quantities of interest for experimentalists to analyze and design. Frequently, for example, an experimentalist is interested in the most probable secondary structure for a set of associated strands to adopt. In this case, rather than a *summation* over all secondary structures, we are looking for a *minimum* over all secondary structures.

To proceed, it is useful to consider the essential logic of the counting problem for secondary structures. At its root, our summation involves the collection of *alternative* secondary structures, each of which is made up of a *composition* of independent loops. Analogously, in calculation of a partition function, the Boltzmann factors for alternative structures are *added* together, while Boltzmann factors of different loops are *multiplied* together. (The summation over structures can be clearly seen in (1.2); the multiplication over loops may be seen by substituting (1.1) into (1.2) and decomposing the exponential.) In algebraic structure terminology, the sum and product operations conducted can be seen to form a *semiring*, that is, a set equipped with two binary operations, both with identity element, and with the second operation distributing over the first. The minimum free energy problem may

thus be solved by replacing the SumProduct semiring with the MinSum semiring, with no change otherwise to the recursions and algorithms. In Chapter 2, we show that this approach works not just for minimum free energies but also for a set of other quantities including overflow-proof partition functions, suboptimal structures, and sampling of structures from the Boltzmann ensemble.

There is a more utilitarian reason to take up the idea of evaluation algebras, which is that it greatly eases the implementation and modularity of software. Generic programming, similarly motivated by abstract algebraic theory [15], plays with a similar idea to our evaluation algebras. The idea of generic programming is to reduce a program to its most essential mathematical structure such that it may be applied to as many uses as possible without modification. Using this technique, we can program the set of model recursions only once, isolated from the operation order targeting each physical quantity. In Chapter 2, by optimizing the other elements of the dynamic programming algorithms in a modular fashion, we are able to achieve large computational speedups for a variety of physical quantities.

1.3 Simulation of secondary structure kinetics

Most analysis and design of nucleic acid secondary structures has been performed in the equilibrium regime. That is, secondary structures are evaluated only via their limiting thermodynamics at an infinite-time horizon. Newer design approaches optimize reaction pathways by considering a set of hypothetical test tubes containing reactants, intermediates, and products [19]; however, true kinetic information is still missing from such an approach. On the other hand, other recent efforts have established preliminary models for kinetics on secondary structures (e.g. [6, 14]). In these models, each secondary structure is treated as an isolated and well-mixed microstate. A system of interacting nucleic acid strands may transition from one secondary structure to another by means of formation or deletion of a single base pair at a time. The rate function is assumed to satisfy detailed balance such that the governing Markov process is recurrent and reversible. Thus, the evolution of $\vec{p}(t)$, the populations of secondary structures at time t, given rate matrix R, follows the matrix ordinary differential equation:

$$\frac{\mathrm{d}\vec{p}(t)}{\mathrm{d}t} = \mathrm{R}^{\top}\,\vec{p}(t). \tag{1.4}$$

1.3.1 Stochastic simulation

Direct solution of nucleic acid kinetics over the entire ensemble of secondary structures is infeasible due to the $e^{O(N)}$ growth in the number of structures with respect to the number of nucleotides N. An alternative to direct simulation of the probability density propagation via (1.4) falls out of the well-known isomorphism between the continuous time Markov chain and its associated discrete-time jump-and-hold chain (e.g. [1]). Simulation of the jump-and-hold chain in biochemical contexts is generally known as the Gillespie algorithm [9]. Using the Gillespie algorithm [9], earlier efforts ([6, 14]) developed an approach for stochastic kinetic simulation of nucleic acid secondary structure trajectories.

The Gillespie method for trajectory simulation is highly optimizable with respect to secondary structure kinetics due to the hierarchical decomposition of interactions enforced by the loop-based model. Exploiting these features, in Chapter 3 we detail the design and implementation of multiple methods for computing trajectories more efficiently. Most of these methods result in substantial improvements to the computational complexity of trajectory simulation with respect to system size. Others result in substantial prefactor improvements, sometimes of an order of magnitude or more. In some cases, we are able to achieve complexity gains by imposing additional constraints on the definition of the microstate rate function. Such cases would indicate that these rate functions should be prioritized in parametrization if possible.

1.3.2 Kinetics including coaxial and dangle stacking contributions

An essential ingredient to performant trajectory simulation is an efficient microstate free energy evaluation. Whereas dynamic programming algorithms efficiently calculate the free energy summing over all possible structures in a complex ensemble, much less attention has been paid to efficiently calculating the free energy of a single structure. This computation becomes significantly more involved when coaxial and dangle stacking states are fully considered. We rectify this situation for the most problematic loop types by applying a transfer-matrix based approach to loop free energy calculation in an ensemble including coaxial and dangle stacking states. Motivated by a similar "state vector" based approach in the lattice path counting literature [2, 10], we use a matrix-trace calculation to incorporate all possible stacking states within a loop, all scaling as $O(n_{\text{bp}})$ in the number of base pairs in the loop n_{bp}.

Most nucleic acid systems of interest involve multiple complexes that may associate or dissociate during a secondary structure trajectory. The calculation of association rates

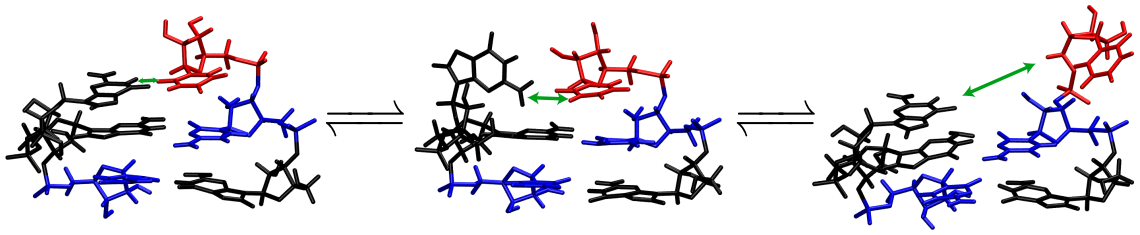


Figure 1.4: Example base pairing reactions simulated using molecular dynamics. Enhanced sampling is used in order to simulate the reaction in which a single base pair is cleaved (here, the topmost base pair). Thermodynamics and kinetics are studied along the reaction pathway from reactant (paired, left) and product (unpaired, right).

between multiple complexes is challenging since the number of possible first contact states grows quadratically in the number of exposed bases in the system. We thus develop a *join propensity* method which exploits certain rate functions to calculate complex join rates in a separable and linear-scaling manner. We adapt the transfer-matrix approach for free energy calculation to the calculation of join propensities including all possible stacking states.

1.4 Computational parametrization of secondary structure models

Secondary structure models are an obvious approximation to the tertiary structures of nucleic acids in nature. In the past, these models have been parametrized over decades of ad hoc melt experiments for RNA and DNA in a limited set of experimental conditions. As secondary structure analysis and design algorithms improve and grow in usage, it is important for the parametrization methods for these models to 1) be as accurate as possible, 2) apply to as many materials as possible (e.g. RNA, DNA, 2′OMe-RNA), 3) apply to as many experimental settings (e.g. temperature, ion concentrations) as possible, and 4) produce kinetic predictions (rather than just equilibrium ones), necessitating completely new sets of parameters and functional forms.

In Chapters 4 and 5, we fulfill these objectives by developing a fully computational workflow based on 1) Gaussian process-based choice of the most informative base pairing reactions to simulate, 2) molecular dynamics (MD) simulations of these reactions (Figure 1.4), and 3) regression of thermodynamic and kinetic information from the yielded trajectories. This effort brings together theoretical techniques from Gaussian process theory with dynamic programming algorithms and molecular dynamics methods.

1.4.1 Simulation design and choice

Molecular dynamics simulations are mature and well-optimized but incur significant computational cost, with research projects commonly requiring millions of CPU hours. For an open-ended study of thermodynamic and kinetic motifs and parameters, it is thus vital to focus on simulations of the reactions that will be most informative to the end objectives. In Chapter 4, by assuming and fitting a Gaussian process over our projected simulation outputs, we develop an approach which maximizes the mutual information of the simulated reaction set $\vec{r}$. In this context, we develop an information gain criterion composed of 1) the covariance matrix $\Gamma_{\vec{r}}$ conditional on the observed simulation outputs and 2) a Fisher information matrix F measuring the sensitivities of complex free energies to each nearest-neighbor parameter. To summarize, we first develop the covariance matrix $\Gamma_{\vec{r}}$ by using the functional forms of the nearest-neighbor free energy model. Next we consider the total decrease in the conditional variance of the complex free energy $\Delta G(\phi)$ over an ensemble of representative sequences ϕ. By linearizing our the complex free energy around its values given empirical parameters, we can pose the total decrease in conditional variance or, equivalently, the mutual information gain, as the following matrix trace:

$$\begin{aligned} \text{information gain} &= \sum_{\phi} \text{Tr}\left[\frac{\partial \Delta G(\phi)}{\partial \vec{\theta}} \Gamma_{\vec{r}} \frac{\partial \Delta G(\phi)}{\partial \vec{\theta}}^{\top}\right] + \text{const} \\ &= \text{Tr}\left[\Gamma_{\vec{r}} \left(\sum_{\phi} \frac{\partial \Delta G(\phi)}{\partial \vec{\theta}}^{\top} \frac{\partial \Delta G(\phi)}{\partial \vec{\theta}}\right)\right] + \text{const} \\ &= \text{Tr}\left[\Gamma_{\vec{r}} F\right] + \text{const.} \end{aligned} \tag{1.5}$$

where the second line in (1.5) follows by the rotational invariance of a matrix trace, and the third line follows from defining the Fisher information matrix as $F \coloneqq \sum_{\phi} \frac{\partial \Delta G(\phi)}{\partial \vec{\theta}}^{\top} \frac{\partial \Delta G(\phi)}{\partial \vec{\theta}}$. Using (1.5) we can efficiently optimize the information gain by automatically tuning the set of reactions $\vec{r}$ to be simulated, as is detailed more in Chapter 4.

1.4.2 Molecular dynamics methodologies

Naïve molecular dynamics simulations of even localized and well-chosen biomolecular reactions can require infeasible timescales of simulation. Classical molecular dynamics simulations typically utilize an $O(1\text{ fs})$ timestep, yielding common trajectory lengths in the $O(1\text{ ns})$ to $O(1\ \mu\text{s})$ range. The timescale of base pairing reactions is only roughly constrained by experimental literature but is known to involve rearrangements on or beyond the slower end of this timescale.

We show that umbrella sampling via application of restraints on collective variable values may be used to provide enhanced sampling of a reaction pathway on a more feasible computational timescale. Applying this approach, we are able to estimate reaction free energies and enthalpies using only short molecular dynamics trajectories.

1.4.3 Thermodynamic and kinetic regression

Secondary structure models should be extensible to different experimental settings, including different temperatures and solvent conditions (including ion concentrations). We thus extend our molecular dynamics methodology to the calculation of reaction enthalpies, allowing approximate extrapolation in temperature, using a reweighted average of internal energies in previously run trajectories. Analogously, we use a Poisson-Boltzmann equation-based approach to accomplish extrapolation of our results to different ion concentrations. Meanwhile, experimentally useful secondary structure models should predict not only reaction free energy but also kinetic information. To meet this goal, we develop a rate constant estimation method based on projection of the molecular dynamics Markov chain down to one dimension, followed by exact solution of rate constants in this reduced chain.

By using the developed Gaussian process statistics, we can optimize free energy and enthalpy parameters of existing nearest-neighbor secondary structure models. We demonstrate in Chapter 5 that we can achieve realistic and relatively accurate thermodynamic parameters using our approach. We also are able to yield some of the first large-scale estimates of individual base pair reaction rate constants. Our approach also yields estimates of free energies, enthalpies, and ion concentration effects.

1.5 Conclusions

In sum, this thesis represents an application of diverse mathematical and chemical methods to simulating and parameterizing nucleic acid secondary structure thermodynamics and kinetics. From the mathematical ingredients of abstract algebras, secondary structure combinatorics, Markov chains, and Gaussian processes, we develop methodologies for efficient dynamic programs, rate calculations, information maximization, and regression of kinetic and thermodynamic parameters. We apply these computational methodologies to both the analysis and parametrization of secondary structure thermodynamic and kinetic models, demonstrating efficient and scalable computational approaches for each of our goals.

BIBLIOGRAPHY

[1] David Aldous and James Fill. *Reversible Markov chains and random walks on graphs*. Berkeley, 1995.

[2] Shaun Ault and Charles Kicey. *Counting lattice paths using Fourier methods*. Springer, 2019.

[3] Robert M. Dirks, Milo Lin, Erik Winfree, and Niles A. Pierce. Paradigms for computational nucleic acid design. *Nucleic Acids Research*, 32(4):1392–1403, 2004.

[4] Robert M. Dirks, Justin S. Bois, Joseph M. Schaeffer, Erik Winfree, and Niles A. Pierce. Thermodynamic analysis of interacting nucleic acid strands. *SIAM Review*, 49(1):65–88, 2007.

[5] Robert Donaghey and Louis W. Shapiro. Motzkin numbers. *Journal of Combinatorial Theory, Series A*, 23(3):291–301, 1977.

[6] Christoph Flamm, Walter Fontana, Ivo L. Hofacker, and Peter Schuster. RNA folding at elementary step resolution. *RNA*, 6(3):325–338, 2000.

[7] Mark E. Fornace, Nicholas J. Porubsky, and Niles A. Pierce. A unified dynamic programming framework for the analysis of interacting nucleic acid strands: Enhanced models, scalability, and speed. *ACS Synthetic Biology*, 9(10):2665–2678, 2020.

[8] Jacques R. Fresco, Bruce M. Alberts, Paul Doty, et al. Some molecular details of the secondary structure of ribonucleic acid. *Nature*, 188:98–101, 1960.

[9] Daniel T. Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *Journal of Computational Physics*, 22(4): 403–434, 1976.

[10] Christian Krattenthaler. Lattice path enumeration. *arXiv preprint arXiv:1503.05930*, 2015.

[11] Gopal Mohanty. *Lattice path counting and applications*. Academic Press, 2014.

[12] Lulu Qian and Erik Winfree. Scaling up digital circuit computation with DNA strand displacement cascades. *Science*, 332(6034):1196–1201, 2011.

[13] Paul W.K. Rothemund. Folding DNA to create nanoscale shapes and patterns. *Nature*, 440(7082):297–302, 2006.

[14] Joseph M. Schaeffer, Chris Thachuk, and Erik Winfree. Stochastic simulation of the kinetics of multiple interacting nucleic acid strands. In *International Workshop on DNA Based Computers*, pages 194–211. Springer, 2015.

[15] Alexander A. Stepanov and Daniel E. Rose. *From mathematics to generic programming*. Pearson Education, 2014.

[16] Gary M. Studnicka, Georgia M. Rahn, Ian W. Cummings, and Winston A. Salser. Computer method for predicting the secondary structure of single-stranded RNA. *Nucleic Acids Research*, 5(9):3365–3388, 1978.

[17] Ignacio Tinoco, Olke C. Uhlenbeck, and Mark D. Levine. Estimation of secondary structure in ribonucleic acids. *Nature*, 230(5293):362–367, 1971.

[18] Leslie G. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3):410–421, 1979.

[19] Brian R. Wolfe, Nicholas J. Porubsky, Joseph N. Zadeh, Robert M. Dirks, and Niles A. Pierce. Constrained multistate sequence design for nucleic acid reaction pathway engineering. *Journal of the American Chemical Society*, 139(8):3134–3144, 2017.

Chapter 2

IMPROVED ALGORITHMS FOR THE EQUILIBRIUM ANALYSIS OF NUCLEIC ACID COMPLEXES

Mark E. Fornace, Nicholas J. Porubsky, and Niles A. Pierce. A unified dynamic programming framework for the analysis of interacting nucleic acid strands: Enhanced models, scalability, and speed. *ACS Synthetic Biology*, 9(10):2665–2678, 2020. doi: 10.1021/acssynbio.9b00523. URL https://doi.org/10.1021/acssynbio.9b00523. PMID: 32910644.

Dynamic programming algorithms within the NUPACK software suite enable analysis of nucleic acid sequences over complex and test tube ensembles containing arbitrary numbers of interacting strand species, serving the needs of researchers in molecular programming, nucleic acid nanotechnology, synthetic biology, and across the life sciences. Here, to enhance the underlying physical model, assure scalability for large calculations, and achieve dramatic speedups when calculating diverse physical quantities over complex and test tube ensembles, we introduce a unified dynamic programming framework that combines three ingredients: 1) recursions that specify the dependencies between subproblems and incorporate the details of the structural ensemble and the free energy model, 2) evaluation algebras that define the mathematical form of each subproblem, 3) operation orders that specify the computational trajectory through the dependency graph of subproblems. The physical model is enhanced using new recursions that operate over the complex ensemble including coaxial and dangle stacking subensembles. The recursions are coded generically and then compiled with a quantity-specific evaluation algebra and operation order to generate an executable for each physical quantity: partition function, equilibrium base-pairing probabilities, MFE energy and structure proxy, suboptimal structures, and Boltzmann sampled structures. For large complexes (e.g., 30,000 nt), scalability is achieved for partition function calculations using an overflow-safe evaluation algebra, and for equilibrium base-pairing probabilities using a backtrack-free operation order. A new blockwise operation order that treats subcomplex blocks for the complex species in a test tube ensemble enables dramatic speedups (e.g., 20–120×) using vectorization and caching. With these performance enhancements, equilibrium analysis of substantial test tube ensembles can be performed in $\leq$ 1 minute on a single computational core (e.g., partition function and equilibrium concentration for all complex species of up to 6 strands formed from 2

strand species of 300 nt each, or for all complex species of up to 2 strands formed from 80 strand species of 100 nt each). A new sampling algorithm simultaneously samples multiple structures from the complex ensemble to yield speedups of an order of magnitude or more as the number of structures increases above $\approx 10^3$. These advances are available within the NUPACK 4.0 code base (`www.nupack.org`) which can be flexibly scripted using the all-new NUPACK Python module.

Dynamic programming algorithms enable efficient and exact equilibrium analysis of nucleic acids with respect to approximate physical models. Algorithms are formulated in terms of nucleic acid secondary structure (i.e., the base pairs of a set of DNA or RNA strands) and employ empirical free energy parameters [4, 16, 17, 19, 20, 22, 24–26, 28, 29, 32, 35] to calculate diverse physical quantities [1, 2, 5, 6, 11, 15, 18, 20, 21, 36]. We have previously developed dynamic programming algorithms that are unique in treating complex and test tube ensembles containing arbitrary numbers of interacting strand species [11], providing crucial tools for capturing concentration effects essential to analyzing and designing the intermolecular interactions that are a hallmark of molecular programming, nucleic acid nanotechnology, and synthetic biology. These algorithms are implemented within NUPACK (Nucleic Acid Package), a growing software suite for the analysis and design of nucleic acid structures, devices, and systems [33].

Here, following 15 years of methods development [8–11, 30, 31, 33, 34], we reconsidered every equilibrium analysis algorithm, arriving at a new unified dynamic programming framework that leads to major improvements of five varieties:

- *Elucidation:* diverse physical quantities are calculated using dynamic programs each combining three ingredients: model-specific recursions, a quantity-specific evaluation algebra, and a quantity-specific operation order.
- *Model:* new recursions capture the structural and energetic details of coaxial and dangle stacking subensembles in the complex ensemble.
- *Scalability:* over-flow safe evaluation algebras and backtrack-free operation orders enable robust partition function and pair probability calculations for large complexes.
- *Speed:* new blockwise operation orders yield dramatic speedups of 1–2 orders of magnitude for equilibrium analysis of test tube ensembles.
- *Brevity:* use of a generic programming paradigm and compile-time polymorphism dramatically reduce the size of the code base.

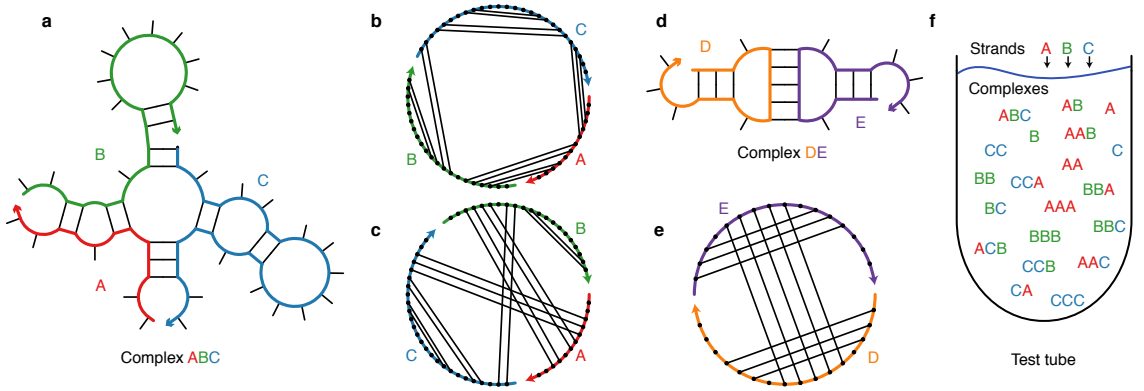


Figure 2.1: Complex and test tube ensembles. (a) A connected unpseudoknotted secondary structure for complex of 3 strands with strand ordering π = ABC. An arrowhead denotes the 3′ end of each strand. (b) Polymer graph representation of the same secondary structure showing no crossing lines for strand ordering π = ABC. (c) Alternative strand ordering π = ACB yields a polymer graph with crossing lines. (c) A pseudoknotted secondary structure with base pairs $i{\cdot}j$ and $d{\cdot}e$ (with $i < d$) that fail to satisfy the nesting property $i < d < e < j$, yielding crossing lines in the corresponding polymer graph (e) for the sole strand ordering π = DE. (f) A test tube ensemble containing strand species Ψ^0 = {A,B,C} interacting to form all complex species Ψ of up to $L_{\max}$ = 3 strands.

We begin by defining the underlying physical model, including definitions of the complex and test tube structural ensembles, and specification of the free energy model for a complex ensemble including coaxial and dangle stacking subensembles. We then describe the unified dynamic programming framework, describing new recursions that capture the details of the enhanced physical model, and new evaluation algebras and operation orders that enable calculation of diverse physical quantities for complex and test tube ensembles of interacting DNA or RNA strands. The resulting suite of algorithms comprise the all-new NUPACK 4.0 analysis code base. Enhanced models, scalability, and speed will benefit researchers in molecular programming, nucleic acid nanotechnology, synthetic biology, and across the life sciences.

2.1 Physical model

2.1.1 Complex ensemble and test tube ensembles

NUPACK algorithms operate over two fundamental ensembles:

- *Complex ensemble:* The ensemble of all (unpseudoknotted connected) secondary structures for an arbitrary number of interacting DNA or RNA strands.
- *Test tube ensemble:* The ensemble of a dilute solution containing an arbitrary number of DNA or RNA strand species (introduced at user-specified concentrations) interacting to form an arbitrary number of complex species.

Furthermore, to enable reaction pathway engineering of dynamic hybridization cascades (e.g., shape and sequence transduction using small conditional RNAs [14]) or large-scale structural engineering including pseudoknots (e.g., RNA origamis [13]), NUPACK generalizes sequence analysis and design to multi-complex and multi-tube ensembles [31].

The sequence, ϕ, of one or more interacting RNA strands is specified as a list of bases $\phi^a \in \{\texttt{A},\texttt{C},\texttt{G},\texttt{U}\}$ for $a = 1, \ldots, |\phi|$. For DNA, $\phi^a \in \{\texttt{A},\texttt{C},\texttt{G},\texttt{T}\}$. A secondary structure, s, of one or more interacting RNA strands is defined by a set of base pairs, each a Watson–Crick pair [$\texttt{A}\cdot\texttt{U}$ or $\texttt{C}\cdot\texttt{G}$] or a wobble pair [$\texttt{G}\cdot\texttt{U}$]. For DNA, the corresponding Watson–Crick pairs are $\texttt{A}\cdot\texttt{T}$ and $\texttt{C}\cdot\texttt{G}$ and there are no wobble pairs. Example secondary structures are displayed in Figures 2.1ad.

For algorithmic purposes, it is convenient to describe secondary structures using a *polymer graph* representation, constructed by ordering the strands around a circle, drawing the backbones in succession from 5′ to 3′ around the circumference with a *nick* between each strand, and drawing straight lines connecting paired bases (e.g., Figure 2.1bc). A secondary structure is *unpseudoknotted* if there exists a strand ordering for which the polymer graph has no crossing lines (e.g., Figure 2.1b), or *pseudoknotted* if all strand orderings contain crossing lines (e.g., the kissing loops of Figure 2.1de). A secondary structure is *connected* if no subset of the strands is free of the others. Consider a *complex* of L distinct strands (e.g., each with a unique identifier in $\{1, \ldots, L\}$) corresponding to strand ordering π. The *complex* ensemble $\overline{\Gamma}(\phi)$ contains all connected polymer graphs with no crossing lines for sequence ϕ and strand ordering π (i.e., all unpseudoknotted secondary structures) [11]. (We dispense with our prior convention [11, 33, 34] of calling this entity an *ordered complex*.)

As a matter of algorithmic necessity, all of the dynamic programs developed in the present work operate on complex ensemble $\overline{\Gamma}(\phi)$ treating all strands as distinct. However, in the laboratory, strands with the same sequence are typically indistinguishable with respect

to experimental observables. Fortunately, for comparison to experimental data, physical quantities calculated over ensemble $\overline{\Gamma}(\phi)$ can be post-processed to obtain the corresponding quantities calculated over ensemble $\Gamma(\phi)$ in which strands with the same sequence are treated as indistinguishable (see Section S5 for details). The ensemble $\Gamma(\phi) \subseteq \overline{\Gamma}(\phi)$ is a maximal subset of distinct secondary structures for strand ordering π. Two secondary structures are indistinguishable if their polymer graphs can be rotated so that all strands are mapped onto indistinguishable strands, all base pairs are mapped onto base pairs, and all unpaired bases are mapped onto unpaired bases; otherwise the structures are distinct [11].

A *test tube* ensemble is a dilute solution containing a set of strand species, Ψ^0, introduced at user-specified concentrations, that interact to form a set of complex species, Ψ, each corresponding to a different strand ordering treating strands with the same sequence as indistinguishable. For L strands, there are $(L-1)!$ strand orderings if all strands are different species (e.g., complexes π = ABC and π = ACB for $L = 3$ and strands A, B, C), but fewer than $(L-1)!$ strand orderings if some strands are of the same species (e.g., complex π = AAA for $L = 3$ with three A strands). By the Representation Theorem of Dirks et al. [11], a secondary structure in the complex ensemble for one strand ordering does not appear in the complex ensemble for any other strand ordering, averting redundancy. It is often convenient to define Ψ to contain all complex species of up to $L_{\max}$ strands (e.g., Figure 2.1f), although Ψ can be defined to contain arbitrary complex species formed from the strand species in Ψ^0.

2.1.2 Loop-based free energy model

For each (unpseudoknotted connected) secondary structure $s \in \overline{\Gamma}(\phi)$, the free energy, $\overline{\Delta G}(\phi, s)$, is estimated as the sum of the empirically determined free energies of the constituent loops [17, 19, 24, 29, 32, 35] plus a strand association penalty[3], ΔG^{assoc}, applied $L-1$ times for a complex of L strands:

$$\overline{\Delta G}(\phi, s) = (L-1)\,\Delta G^{\text{assoc}} + \sum_{\text{loop}\in s} \Delta G(\text{loop}). \tag{2.1}$$

The secondary structure and polymer graph of Figure 2.2 illustrate the different loop types, with free energies modeled as follows [17, 19, 24, 29, 32, 35]:

- A *hairpin loop* is closed by a single base-pair $i \cdot j$. The loop free energy, $\Delta G^{\text{hairpin}}_{i,j}$, depends on sequence and loop size.
- An *interior loop* is closed by two base pairs ($i \cdot j$ and $d \cdot e$ with $i < d < e < j$). The loop free energy, $\Delta G^{\text{interior}}_{i,d,e,j}$ depends on sequence, loop size, and loop asymmetry.

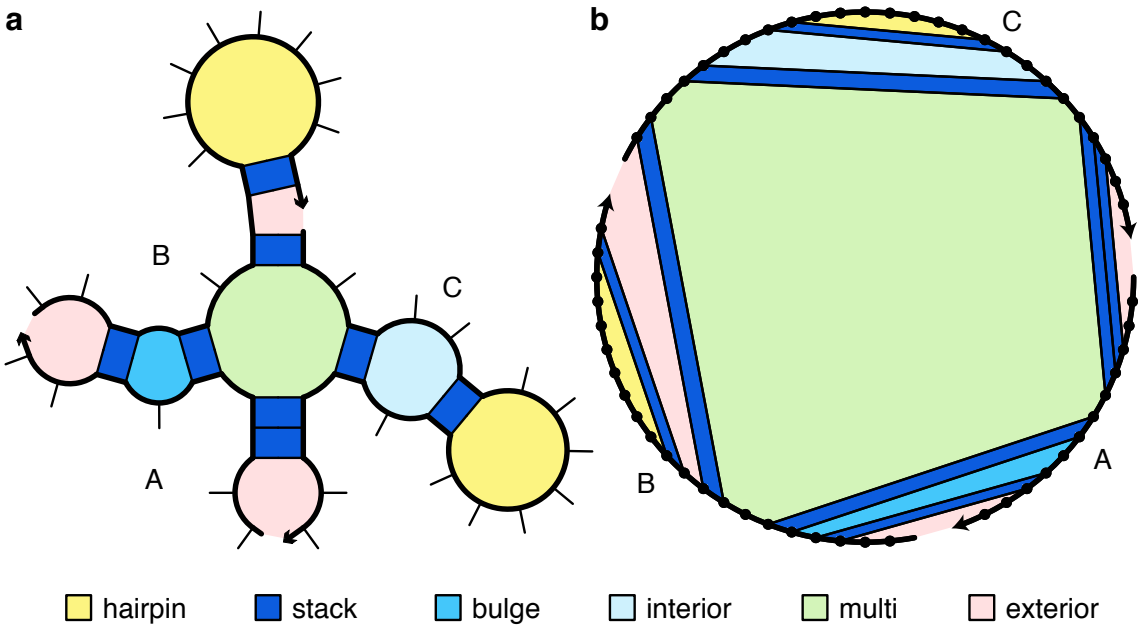


Figure 2.2: Loop-based free energy model for a complex. (a) Canonical loop types for complex with strand ordering π = ABC. (b) Equivalent polymer graph representation. An arrowhead denotes the 3′ end of each strand.

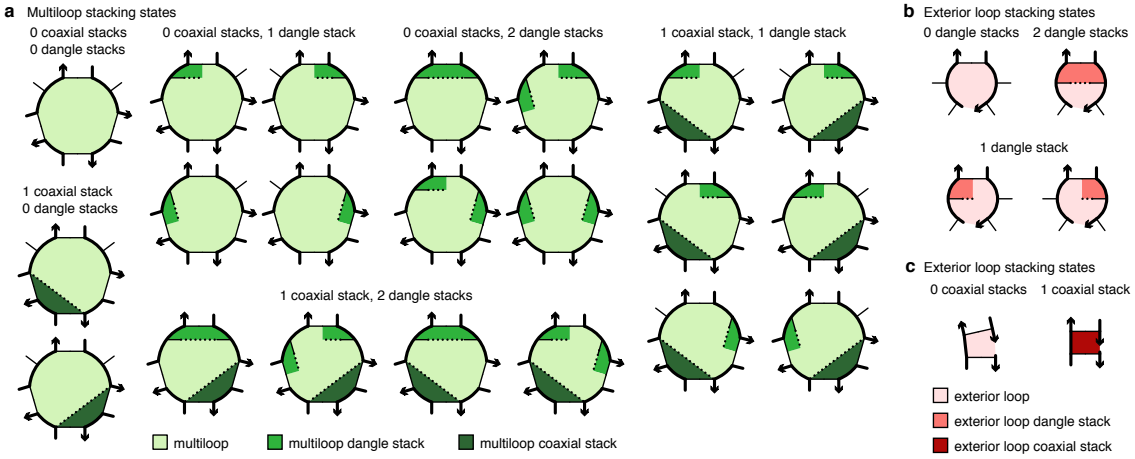


Figure 2.3: Coaxial and dangle stacking states for multiloops and exterior loops. (a) Stacking subensemble for the multiloop of Figure 2.2a. (b,c) Stacking subensembes for two exterior loops from Figure 2.2a.

Bulge loops (where either $d = i + 1$ or $e = j - 1$) and *stacked pairs* (where both $d = i + 1$ and $e = j - 1$) are treated as special cases of interior loops.

- A *multiloop* is closed by three or more base pairs. The loop free energy is modeled as the sum of three sequence-independent penalties: (1) $\Delta G^{\text{multi}}_{\text{init}}$ for formation of a multiloop, (2) $\Delta G^{\text{multi}}_{\text{bp}}$ for each closing base pair, (3) $\Delta G^{\text{multi}}_{\text{nt}}$ for each unpaired

nucleotide inside the multiloop, plus a sequence-dependent penalty: (4) $\Delta G_{i,j}^{\text{terminalbp}}$ for each closing pair $i \cdot j$.

- An *exterior loop* contains a nick between strands and any number of closing base pairs. The exterior loop free energy is the sum of $\Delta G_{i,j}^{\text{terminalbp}}$ over all closing base pairs $i \cdot j$. Hence, an unpaired strand has a free energy of zero, corresponding to the reference state [11].

2.1.3 Coaxial and dangle stacking subensembles within complex ensembles

Within a multiloop or an exterior loop, there is a subensemble of coaxial stacking states between adjacent closing base pairs and dangle stacking states between closing base pairs and adjacent unpaired bases. The physical model for multiloops and exterior loops has previously been enhanced for the ensemble of a single strand [20] by incorporating coaxial stacking [19, 22, 29] and dangle stacking [4, 26, 29, 35] terms into the multiloop and exterior loop free energies. For the complex ensemble, we have previously neglected coaxial stacking and incorporated a heuristic dangle stacking state [11]. Here, we exactly incorporate all coaxial and dangle stacking states in the complex ensemble. Within a multiloop or exterior loop, a base pair can form one *coaxial stack* with an adjacent base pair, or can form a *dangle stack* with at most two adjacent unpaired bases; unpaired bases can either form no stack, or can form a dangle stack with at most one adjacent base pair. See Figure 2.3 for an illustration of the valid stacking states for a multiloop (panel a) or two exterior loops (panels b and c).

For a given multiloop or exterior loop, the energetic contributions of all possible coaxial and dangle stacking states are enumerated so as to calculate the free energy:

$$\Delta G^{\text{stacking}} = -kT \log \sum_{\omega \in \text{loop}} \prod_{x \in \omega} e^{-\Delta G_x / kT} \tag{2.2}$$

where ω indexes the possible stacking states within the loop and x indexes the individual stacks (coaxial or dangle) within a stacking state. The free energy of a multiloop or exterior loop is augmented by the corresponding $\Delta G^{\text{stacking}}$ bonus. Hence, a secondary structure s continues to be defined as a set of base pairs, and the stacking states within a given multiloop or exterior loop are treated as a structural subensemble that contributes in a Boltzmann-weighted fashion to the free energy model for the loop. Let $s^{||} \in s$ denote a stacking state of the paired and unpaired bases in s. We may equivalently define the free energy of secondary structure s in terms of the free energies for all stacking states $s^{||} \in s$:

$$\overline{\Delta G}(\phi, s) = -kT \log \sum_{s^{||} \in s} e^{-\overline{\Delta G}(\phi, s^{||})/kT}. \tag{2.3}$$

2.1.4 Symmetry correction

Let $\overline{\Gamma}^{||}(\phi)$ denote the ensemble of stacking states corresponding to the complex ensemble of secondary structures $\overline{\Gamma}(\phi)$. For a secondary structure $s \in \Gamma(\phi)$ with an R-fold rotational symmetry there is in R-fold reduction in distinguishable conformational space, so the free energy (2.1) must be adjusted[11] by a symmetry correction:

$$\Delta G(\phi, s) = \overline{\Delta G}(\phi, s) + \Delta G^{\text{sym}}(\phi, s). \tag{2.4}$$

where

$$\Delta G^{\text{sym}}(\phi, s) = kT \log R(\phi, s). \tag{2.5}$$

Because the symmetry factor $R(\phi, s)$ is a global property of each secondary structure $s \in \Gamma(\phi)$, it is not suitable for use with dynamic programs that treat multiple subproblems simultaneously without access to global structural information. As a result, dynamic programs operate on ensemble $\overline{\Gamma}(\phi)$ using physical model (2.1) and then the Distinguishability Correction Theorem of Dirks et al. [11] enables exact conversion of physical quantities to ensemble $\Gamma(\phi)$ using physical model (2.4). Interestingly, ensembles $\overline{\Gamma}(\phi)$ and $\Gamma(\phi)$ both have utility when examining the physical properties of a complex as they provide related but different perspectives, akin to complementary thought experiments (see Section S5).

2.1.5 Free energy parameters

Supported temperature-dependent RNA and DNA parameter sets include:

- `rna95` based on (Serra & Turner, 1995)[26] with additional parameters[35] including coaxial stacking [19, 29] and dangle stacking [26, 29, 35] in 1M Na^+.
- `dna04` based on (SantaLucia, 1998)[24] and (SantaLucia & Hicks, 2004)[25] with additional parameters[35] including coaxial stacking [22] and dangle stacking [4, 35] in user-specified concentrations of Na^+, K^+, NH_4^+, and Mg^{++} (see Section S1.2 for details on implementation of the salt corrections)[16, 22, 24, 25].
- `rna06` based on (Mathews et al., 1999)[19], (Mathews et al., 2004)[20], and (Lu et al., 2006)[17] with additional parameters[32, 35] including coaxial stacking [19, 29] and dangle stacking [26, 29, 35] in 1M Na^+.
- `custom` using user-specified parameters representing nucleic acids or synthetic nucleic acid analogs in experimental conditions of choice.

See Sections S1.5–S1.7 for details about parameter sets.

2.2 Algorithm

2.2.1 Physical quantities

Consider a complex with sequence ϕ. We provide dynamic programs to calculate:

- the partition function,

$$\overline{Q}(\phi) = \sum_{s \in \overline{\Gamma}(\phi)} e^{-\overline{\Delta G}(\phi,s)/kT}, \tag{2.6}$$

  over ensemble $\overline{\Gamma}(\phi)$ treating all strands as distinct. The equilibrium probability of any secondary structure $s \in \overline{\Gamma}(\phi)$ is then

$$\overline{p}(\phi, s) = e^{-\overline{\Delta G}(\phi,s)/kT} / \overline{Q}(\phi). \tag{2.7}$$

  Post-processing $\overline{Q}(\phi)$ yields the partition function $Q(\phi)$ over ensemble $\Gamma(\phi)$ treating strands with the same sequence as indistinguishable[11].
- the base-pairing probability matrix $\overline{P}(\phi)$ with entries $\overline{P}^{i,j}(\phi) \in [0, 1]$ corresponding to the probability

$$\overline{P}^{i,j}(\phi) = \sum_{s \in \overline{\Gamma}(\phi)} \overline{p}(\phi, s) S^{i,j}(s) \tag{2.8}$$

  that base pair $i \cdot j$ forms at equilibrium within ensemble $\overline{\Gamma}(\phi)$, treating all strands as distinct. Here, $S(s)$ is a *structure matrix* with entries $S^{i,j}(s) = 1$ if structure s contains base pair $i \cdot j$ and $S^{i,j}(s) = 0$ otherwise. Abusing notation, the entry $S^{i,i}(s)$ is 1 if base i is unpaired in structure s and 0 otherwise; the entry $\overline{P}^{i,i}(\phi) \in [0, 1]$ denotes the equilibrium probability that base i is unpaired over ensemble $\overline{\Gamma}(\phi)$. Hence $S(s)$ and $\overline{P}(\phi)$ are symmetric matrices with row and column sums of 1.
- the free energy of the minimum free energy (MFE) stacking state $s^{||}_{\text{MFE}}(\phi) \in \overline{\Gamma}^{||}(\phi)$ treating all strands as distinct:

$$\overline{\Delta G}(\phi, s^{||}_{\text{MFE}}) = \min_{s^{||} \in \overline{\Gamma}^{||}(\phi)} \overline{\Delta G}(\phi, s^{||}). \tag{2.9}$$

- the MFE proxy structure

$$s_{\text{MFE}'} = \{s \in \overline{\Gamma}(\phi) | s^{||}_{\text{MFE}} \in s, s^{||}_{\text{MFE}}(\phi) = \arg \min_{s^{||} \in \overline{\Gamma}^{||}(\phi)} \overline{\Delta G}(\phi, s^{||})\}. \tag{2.10}$$

  defined as the secondary structure containing the MFE stacking state within its subensemble. If there is more than one MFE stacking state, the algorithm returns all corresponding MFE proxy structures.

- the set of suboptimal secondary structures

$$\overline{\Gamma}_{\text{subopt}}(\phi, \Delta G_{\text{gap}}) = \{s \in \overline{\Gamma}(\phi) | s^{\text{II}} \in s, \overline{\Delta G}(\phi, s^{\text{II}}) \leq \overline{\Delta G}(\phi, s^{\text{II}}_{\text{MFE}}) + \Delta G_{\text{gap}}\} \tag{2.11}$$

  with stacking states within a specified $\Delta G_{\text{gap}} \geq 0$ of the MFE stacking state.

- a set of J secondary structures Boltzmann sampled from ensemble $\overline{\Gamma}(\phi)$ treating all strands as distinct:

$$\overline{\Gamma}_{\text{sample}}(\phi, J) \in \overline{\Gamma}(\phi). \tag{2.12}$$

  Post-processing then yields the set of J secondary structures Boltzmann sampled from ensemble $\Gamma(\phi)$ treating strands with the same sequence as indistinguishable:

$$\Gamma_{\text{sample}}(\phi, J) \in \Gamma(\phi). \tag{2.13}$$

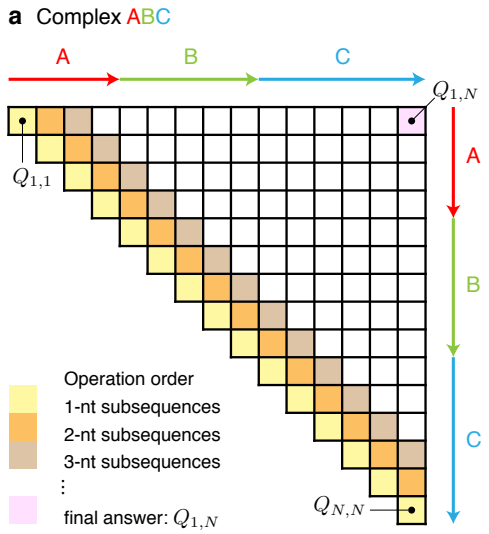


Figure 2.4: Operation order for partition function dynamic program over a complex ensemble with N nucleotides.

Now consider a test tube ensemble containing an arbitrary set of strand species Ψ^0 interacting to form an arbitrary set of complex species Ψ. We provide algorithms to calculate:

- the set of equilibrium concentrations $x_\Psi \equiv x_c \ \forall c \in \Psi$, (specified as mole fractions) that are the unique solution to the strictly convex optimization problem[11]:

$$\min_{x_\Psi} \sum_{c \in \Psi} x_c (\log x_c - \log Q_c - 1) \tag{2.14a}$$

$$\text{subject to} \quad \sum_{c \in \Psi} A_{i,c} x_c = x_i^0 \quad \forall i \in \Psi^0, \tag{2.14b}$$

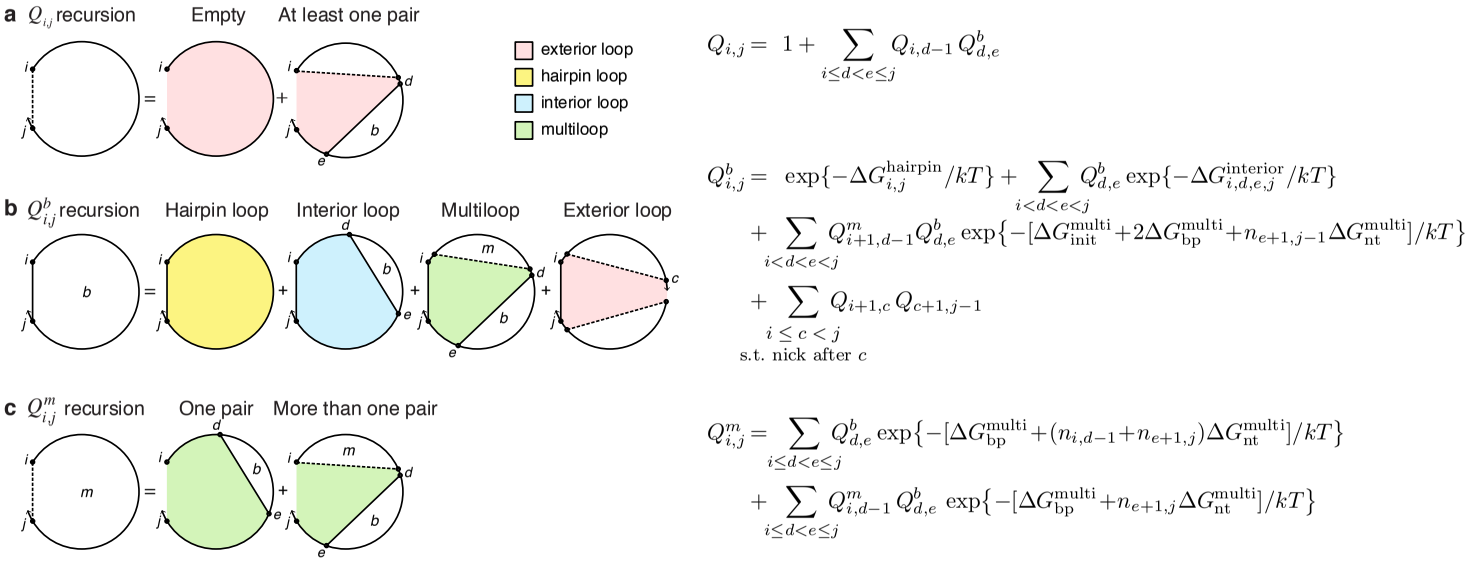


Figure 2.5: Partition function dynamic program recursion diagrams (left) and recursion equations (right) [11]. A solid straight line indicates a base pair and a dashed line demarcates a region without implying that the connected bases are paired. Shaded regions correspond to loop free energies that are explicitly incorporated at the current level of recursion (colors correspond to the loop types of Figure 2.2). (a) $Q_{i,j}$ represents the partition function for subsequence $[i, j]$. There are two cases: either there are no base pairs (corresponding to the reference free energy 0 and partition function contribution 1) or there is a 3′-most base pair $d \cdot e$. In the latter case, determination of the partition function contribution makes use of previously computed subsequence partition functions $Q^b_{d,e}$ and $Q_{i,d-1}$. By the distributive law, multiplication of these subsequence partition functions (each representing a sum over substructures) implicitly sums over all pairwise combinations of substructures. The independence of the loop contributions in the energy model (2.1) implies that these products appropriately add the free energies in the exponents. (b) $Q^b_{i,j}$ is the partition function for subsequence $[i, j]$ with the restriction that bases i and j are paired. There are four cases: either there are no additional base pairs (corresponding to a hairpin loop), there is exactly one additional base pair $d \cdot e$ (corresponding to an interior loop), there is more than one additional base pair (corresponding to a multiloop) with 3′-most pair $d \cdot e$ and at least one additional pair specified in a previously computed subsequence partition function $Q^m_{i,d-1}$, or there is an exterior loop containing a nick after nucleotide c. $n_{i,j} \equiv j - i + 1$ denotes the number of nucleotides between i and j inclusive. (c) $Q^m_{i,j}$ is the partition function for subsequence $[i, j]$ with the restrictions that the subsequence is inside a multiloop and contains at least one base pair. There are two cases: either there is exactly one additional base pair $d \cdot e$ defining the multiloop, or there is more than one additional base pair defining the multiloop (with 3′-most pair $d \cdot e$). The time complexity of these recursions is $O(N^4)$ (indices i, d, e, j on the right hand side) and the space complexity is $O(N^2)$ (indices i, j on the left hand side).

expressed in terms of the previously calculated set of partition functions Q_Ψ. Here, the constraints impose conservation of mass: A is the stoichiometry matrix such that $A_{i,c}$ is the number of strands of type i in complex c, and x_i^0 is the total concentration of strand i present in the test tube. Based on dimensional analysis [11], the convex optimization algorithm operates on mole fractions, but for convenience, accepts molar strand concentrations $[i]^0 = x_i^0 \rho_{\mathrm{H_2O}}$ as inputs and returns molar complex concentrations $[c] = x_c \rho_{\mathrm{H_2O}}$ as outputs, where $\rho_{\mathrm{H_2O}}$ is the molarity of water.

- the ensemble pair fractions for the test tube ensemble, for example

$$f_A(i_A \cdot j_B) \tag{2.15}$$

denotes the fraction of A strands that form base pair $i_A \cdot j_B$ (correspondingly $f_B(i_A \cdot j_B)$ denotes the fraction of B strands that form base pair $i_A \cdot j_B$). In order to calculate these base-pairing observables, it is first necessary to calculate the set of equilibrium concentrations x_Ψ and the set of base-pairing probability matrices $\overline{P}_\Psi$.

Table 2.1: Algorithmic ingredients for calculating diverse physical quantities.

Quantity	Symbol	Recursions	Evaluation Algebra	Dependency	Operation Order
Partition function	$\overline{\underline{Q}}(\phi)$	Stacking	SumProduct, SplitExp	–	Blockwise forward sweep
MFE	$\overline{\Delta G}(\phi, s^{\shortparallel}_{\text{MFE}})$	Stacking	MinSum	–	Blockwise forward sweep
Complex ensemble size	$\lvert\overline{\Gamma}(\phi)\rvert$	No stacking	Count	–	Blockwise forward sweep
Pair probability matrix	$\overline{P}(\phi)$	Stacking	SumProduct, SplitExp	–	Blockwise forward sweep
Sampled ensemble	$\overline{\Gamma}_{\text{sample}}(\phi, J)$	Stacking	ArgRandJ	$\overline{\underline{Q}}(\phi)$	Backtracking, priority queue
MFE structure proxy	$s_{\text{MFE}'}(\phi)$	Stacking	ArgMin	$\overline{\Delta G}(\phi, s^{\shortparallel}_{\text{MFE}})$	Backtracking, stack
Suboptimal ensemble	$\overline{\Gamma}_{\text{subopt}}(\phi, \Delta G_{\text{gap}})$	Stacking	ArgMinGap	$\overline{\Delta G}(\phi, s^{\shortparallel}_{\text{MFE}})$	Backtracking, stack
Concentrations	x_Ψ	–	–	Q_Ψ	Convex optimization
Ensemble pair fractions	$f_A(i_A \cdot j_B)$	–	–	$x_\Psi, \overline{P}_\Psi$	–

2.2.2 Existing dynamic programs

Before describing the new unified dynamic programming framework, it is helpful to briefly summarize existing algorithms that operate on complex ensemble $\overline{\Gamma}(\phi)$ using a simplified free energy model that neglects coaxial stacking and approximates dangle stacking [11]. The complex ensemble size, $|\overline{\Gamma}(\phi)|$, grows exponentially with the number of nucleotides (Figure S37), $N \equiv |\phi|$, but the partition function can be calculated in $O(N^3)$ time and $O(N^2)$ space using a dynamic program [11, 21]. The algorithm calculates the subsequence partition function $Q_{i,j}$ for each subsequence $[i, j]$ via a forward sweep from short subsequences to the full sequence (Figure 2.4), finally yielding the partition function of the full sequence, $Q_{1,N}$. The recursions used to calculate $Q_{i,j}$ from previously calculated subsequence partition functions can be depicted as recursion diagrams (Figure 2.5 left; with free energy contributions colored to match the loop types of Figure 2.2) or equivalently using recursion equations (Figure 2.5 right). The Q recursion relies on additional restricted partition functions Q^b and Q^m that are also calculated recursively. Collectively, the Q, Q^b, and Q^m recursions yield $\overline{Q}(\phi) = Q_{1,N}$, incorporating the partition function contributions of every structure $s \in \overline{\Gamma}(\phi)$ based on free energy model (2.1) treating all strands as distinct. After calculating the partition function with a forward sweep from short to long sequences, dynamic programs that backtrack through the matrix of subsequence partition functions from long to short subsequences can be used to calculate the matrix of equilibrium base-pairing probabilities, $\overline{P}(\phi)$, [9, 11, 21] or to Boltzmann sample a structure from ensemble $\overline{\Gamma}(\phi)$ [6, 11].

The partition function dynamic program can be converted into an MFE dynamic program in a straightforward way by replacing every product of exponentiated free energies with a sum of free energies and every sum of alternative partition function contributions with a minimization over alternative free energy contributions, yielding the MFE of the full sequence, $\overline{\Delta G}(\phi, s_{\text{MFE}}) = F_{1,N}$ [11, 36]. After calculating the MFE with a forward sweep from short to long subsequences, dynamic programs that backtrack through the matrix of subsequence MFEs from long to short subsequences can be used to determine the MFE secondary structure(s), $s_{\text{MFE}}(\phi) \in \overline{\Gamma}(\phi)$, or the ensemble of suboptimal structures, $\overline{\Gamma}_{\text{subopt}}(\phi, \Delta G^{\text{gap}})$. At the heart of the improvements in the present work is a new unified treatment of this suite of dynamic programs for calculating diverse physical quantities.

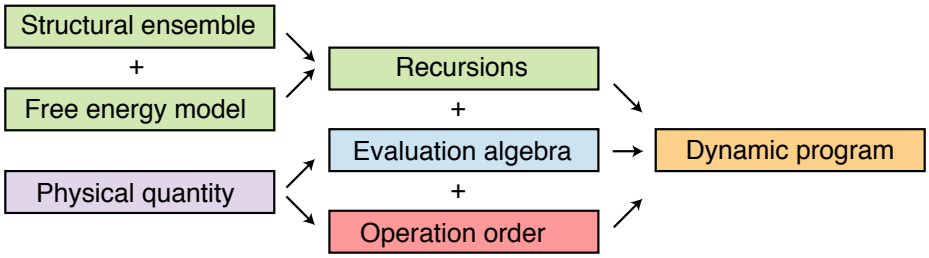


Figure 2.6: Unified dynamic programming framework. To calculate a physical quantity of interest based on a physical model comprising a structural ensemble and a free energy model, each dynamic program combines three ingredients: model-specific recursions, a quantity-specific evaluation algebra, and a quantity-specific operation order.

2.2.3 Unified dynamic programming framework

In the new unified framework, each dynamic program combines three ingredients (Figure 2.6): a set of recursions, an evaluation algebra, and an operation order. A set of recursions specifies the dependencies of each subproblem, capturing the structural details of the complex ensemble and the energetic details of the loop-based free energy model. An evaluation algebra yields the mathematical form of each subproblem, allowing recursions to be generically extended to each physical quantity of interest. An operation order defines the computational trajectory through the dependency graph of subproblems, yielding dramatic speedups using appropriate data structures. In the following sections, we first introduce a new set of recursions that treat the enhanced physical model including coaxial and dangle stacking, and then describe evaluation algebras and operation orders that enable calculation of diverse physical quantities for complex and test tube ensembles (Table 2.1).

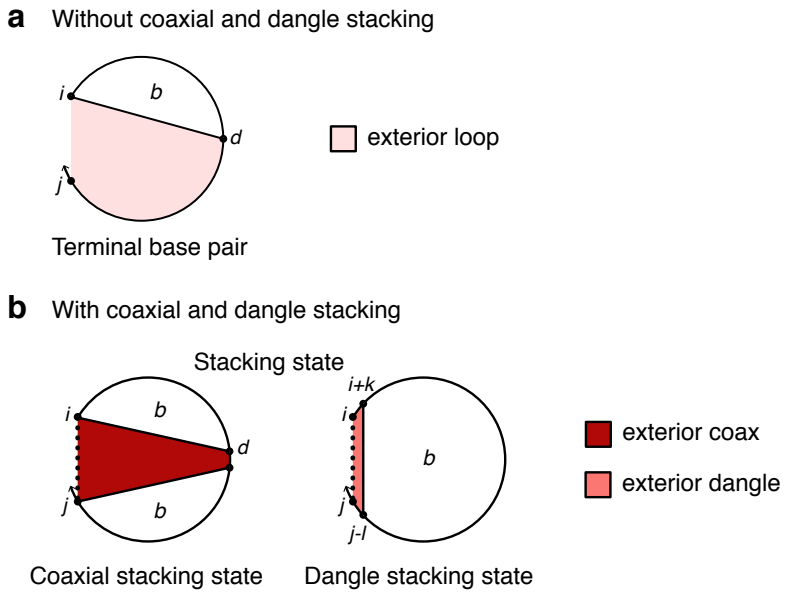


Figure 2.7: Elementary recursion entities without or with coaxial and dangle stacking. (a) Terminal base pair: a base pair that terminates a duplex ($i \cdot d$) in an exterior loop or multiloop context. (b) Stacking state: (1) Either a coaxial stacking state: two adjacent terminal base pairs that are coaxially stacked ($i \cdot d$ and $d + 1 \cdot j$) in an exterior loop or multiloop context, (2) or a dangle stacking state: zero, one, or two unpaired nucleotides (neither i nor j, i only, j only, both i and j) dangle stacking on an adjacent terminal base pair ($i + k \cdot j - l$) in an exterior loop or multiloop context. Shading denotes free energies incorporated by the recursion.

2.2.4 Recursions for the complex ensemble with coaxial and dangle stacking

To treat the enhanced physical model including coaxial and dangle stacking contributions for all multiloops and exterior loops, we require a new set of recursions that incorporate the subensemble of stacking states and free energies defined by equation (2.2) and illustrated in Figure 2.3. For the recursions without coaxial and dangle stacking, the elementary recursion entity is a *terminal base pair* (Figure 2.7a; a base pair that terminates a duplex in an exterior loop or multiloop context). For example, a recursion might contain exactly one terminal base pair, a 3′-most terminal base pair, or one or more terminal base pairs. By contrast, for new recursions with coaxial and dangle stacking, the elementary recursion entity becomes a *stacking state* (Figure 2.7b), which may be either a coaxial stacking state (two adjacent terminal base pairs that are coaxially stacked in a multiloop or exterior loop context), or a dangle stacking state (zero, one, or two unpaired nucleotides dangle stacking on an adjacent terminal base pair in a multiloop or exterior loop context). For example, a recursion might contain exactly one stacking state, a 3′-most stacking state, or one or more stacking states. Note that a terminal base pair without coaxial and dangle stacking

corresponds to the subset of a dangle stacking state where there are zero nucleotides dangle stacking, so the complex ensemble without coaxial and dangle stacking is a subset of the complex ensemble with coaxial and dangle stacking. The inclusion of coaxial and dangle stacking subensembles adds significant complexity to the specification of recursions. The full set of $O(N^3)$ recursions with coaxial and dangle stacking are provided in Section S2. In the following sections, we describe how diverse physical quantities can be calculated using these recursions in combination with different evaluation algebras and operation orders.

	Algebra	Algorithm Output	$\mathbb{0}$	$\mathbb{1}$	$a \oplus b$	$a \otimes b$	$W(g)$
a	SumProduct	Partition function	0	1	$a + b$	$a \cdot b$	$e^{-g/kT}$
	Count	Ensemble size	0	1	$a + b$	$a \cdot b$	1
	MinSum	MFE	∞	0	$\min(a, b)$	$a + b$	g
b	SplitExp	Partition function					
	Mantissa		0	1	$a_{\mathrm{m}} \cdot 2^{a_{\mathrm{e}}+\gamma} + b_{\mathrm{m}} \cdot 2^{b_{\mathrm{e}}+\gamma}$	$a_{\mathrm{m}} \cdot b_{\mathrm{m}}$	$e^{-g/kT}$
	Exponent		0	γ	0	$a_{\mathrm{e}} + b_{\mathrm{e}} + \gamma$	γ
c	ArgRand	Sampled structure					
	Value		0	1	$a_{\mathrm{v}} + b_{\mathrm{v}}$	$a_{\mathrm{v}} \cdot b_{\mathrm{v}}$	$e^{-g/kT}$
	Elements		$\varnothing$	$\varnothing$	$\arg\,\mathrm{rand}(a_{\mathrm{v}}, b_{\mathrm{v}})$	$a_\lambda \cup b_\lambda$	$\varnothing$
d	ArgMin	MFE structure proxy					
	Value		∞	0	$\min(a_{\mathrm{v}}, b_{\mathrm{v}})$	$a_{\mathrm{v}} + b_{\mathrm{v}}$	g
	Elements		$\varnothing$	$\varnothing$	$\arg\min(a_{\mathrm{v}}, b_{\mathrm{v}})$	$a_\lambda \cup b_\lambda$	$\varnothing$

Table 2.2: Evaluation algebras for dynamic programming algorithms operating on a complex ensemble. a and b are elements within a given evaluation algebra domain. SumProduct yields the partition function of the complex ensemble. Count yields the number of secondary structures in the complex ensemble. MinSum yields the free energy of the MFE stacking state in the complex ensemble. SplitExp yields the partition function in split mantissa/exponent form using a given exponent shift γ in order to avoid overflow for the complex ensemble. ArgRand yields a Boltzmann sampled secondary structure with partition function value x_{v} and associated with recursion elements x_λ. ArgMin yields the secondary structure containing the MFE stacking state with free energy value x_{v} and associated with recursion elements x_λ. See Section *S*3 for details.

2.2.5 Evaluation algebras for partition function, minimum free energy, and ensemble size

As previously noted for the complex ensemble without coaxial and dangle stacking, the partition function recursion diagrams of Figure 2.5a can equivalently be expressed as the

partition function recursion equations of Figure 2.5b, and these in turn can be systematically transformed into recursion equations to calculate the MFE. Alternatively, we may view the partition function and MFE recursion equations as the results of applying two different evaluation algebras to a generic set of recursion diagrams and equations that capture the details of a given physical model (comprising a structural ensemble and a free energy model). Here, we formalize an *evaluation algebra* as an algebraic structure composed of: 1) a semiring R equipped with commutative binary operators $\oplus$ and $\otimes$ and associated identity elements $\mathbb{0}$ and $\mathbb{1}$, 2) a map W from free energy parameters to R with the property $W(0) = \mathbb{1}$, and 3) a map Q from recursion indices to R. Table 2.2a defines the evaluation algebras for the partition function and MFE algorithms, as well as the evaluation algebra for calculating the size of the complex ensemble, $|\overline{\Gamma}(\phi)|$. For example, using the SumProduct evaluation algebra to calculate the partition function: 1) $\oplus$ is standard addition, $\otimes$ is standard multiplication, $\mathbb{0}$ is 0, $\mathbb{1}$ is 1, 2) $W(g)$ is the Boltzmann factor $\exp(-g/kT)$ with the property $W(0) = 1$, and 3) Q is the trivial matrix lookup operator $Q(n, i, j) \mapsto Q^n_{i,j}$, where n denotes the type of recursion (e.g., $n = b$ for a Q^b recursion). The evaluation algebras used to calculate the partition function, MFE, and complex ensemble size can be applied to recursions that operate over the complex ensemble with or without coaxial and dangle stacking subensembles.

This paradigm of applying a quantity-specific evaluation algebra to a model-specific set of recursions extends to diverse physical quantities, as we describe in the sections that follow. This generic programming abstraction dramatically reduces the size of the code base and enforces implementation correctness. Instead of writing separate code to upgrade the recursion equations to the new physical model for each physical quantity, a single set of recursion equations is coded and compiled using C++ expression templates for each of the evaluation algebras in Table 2.2 to produce a suite of executables for calculating the corresponding physical quantities.

2.2.6 Overflow-safe evaluation algebra for large partition function calculations

One of the challenges with calculating the partition function is the prevention of overflow as the size of the complex, $N \equiv |\phi|$, increases. Using double-precision (64-bit) arithmetic, the maximum expressible number is $\approx 10^{308}$, enabling calculation of partition functions for complexes of ≈1400 nt for random sequences and ≈450 nt for designed sequences (which typically have a free energy landscape with a deep well). Using quadruple-precision (128-bit) arithmetic, the maximum expressible number is increased to $\approx 10^{4932}$ (platform-dependent), which enables partition function calculations for complexes of up to ≈22,000 nt

for random sequences and ≈7000 nt for designed sequences (at the cost of doubled storage) [11].

Here, to enable partition function calculations for even larger complexes, we define an overflow safe evaluation algebra that operates separately on the mantissa and exponent for the partition function calculation (Table 2.2b). The elements of the partition function recursion matrix are represented as $a = a_\mathrm{m} 2^{a_\mathrm{e}}$, where a_m is a single-precision (32-bit) float and a_e is a 32-bit integer, so the maximum expressible number is $\approx 10^{646457031}$.

For exposition, we assume in Table 2.2 that any expression is to be calculated with respect to a known reference exponent shift, γ, to which the expression is aligned. For instance, consider the expression $a \otimes b$ where $a = 40$ ($a_\mathrm{m} = 0.625$, $a_\mathrm{e} = 6$), $b = 96$ ($b_\mathrm{m} = 0.75$, $b_\mathrm{e} = 7$), and $\gamma = -6$, then $x_\mathrm{m} = a_\mathrm{m} \cdot b_\mathrm{m} = 0.625 \cdot 0.75 = 0.46875$ and $x_\mathrm{e} = a_\mathrm{e} + b_\mathrm{e} + \gamma = 6 + 7 - 6 = 7$ corresponding to $a \otimes b = x_\mathrm{m} \cdot 2^{x_\mathrm{e}} \cdot 2^{-\gamma} = 0.46875 \cdot 2^7 \cdot 2^6 = 0.46875 \cdot 2^{13}$. The recursion result may thus be calculated and stored as (0.46875,13) without explicitly computing its real equivalent, 3840. See Section S3.1.4 for a full description of the evaluation algebra including selection of an appropriate γ for each expression.

With this construction, the storage cost is thus identical to using double-precision but overflow is no longer limiting, and the space and time complexity of the algorithm become the limiting factors. Empirically, we observe a ≈2–2.5× increase in cost for the overflow-safe evaluation algebra relative to a double-precision floating point evaluation algebra (Figure S41). In practice, we use a blended approach by switching between the single-precision SumProduct, double-precision SumProduct, and the SplitExp evaluation algebras as overflow occurs during the partition function calculation for a given complex.

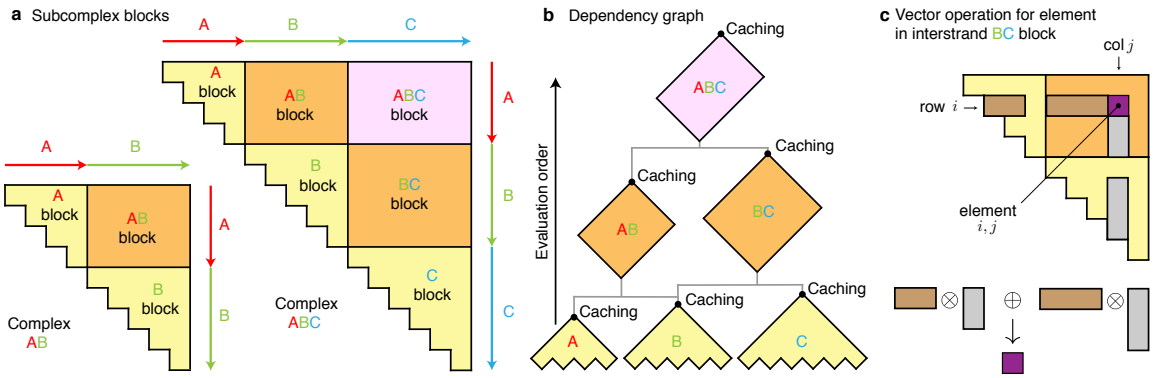


Figure 2.8: Blockwise operation order for dynamic programs operating on complex and test tube ensembles. (a) Subcomplex blocks within dynamic programming matrices (cf. Figure 2.4): triangular intrastrand blocks (A, B, C) and rectangular interstrand blocks (AB, BC, ABC) for complexes AB and ABC. Element i, j corresponds to a conditional ensemble for subsequence $[i, j]$ which contains no nicks if i, j is in an intrastrand block and one or more nicks if i, j is in an interstrand block. (b) Dependency graph for block evaluation: bottom to top for forward algorithms (depicted), top to bottom for backtracking algorithms. (c) Each recursion operation for calculation of element i, j in an interstrand block (e.g., $Q_{i,j} \leftarrow \sum_{i \le d < j} Q_{i,d} Q_{d+1,j}$) can be implemented as multiple vectorized dot products between valid subvectors of row i (brown) and valid subvectors of column j (gray) to obtain element i, j (purple), where valid positions are those that avoid introducing disconnected structures into the complex ensemble.

2.2.7 Efficient blockwise dynamic programs over subcomplexes using caching and vectorization

To this point, we have considered dynamic programs that operate on a complex of L strands. We now re-examine that goal in the more general context of a test tube ensemble containing the set of strand species Ψ^0 interacting to form the set of complex species Ψ. For example, suppose Ψ^0 contains M strand species and Ψ is defined to contain all complexes of up to $L_{\max}$ strands. The simplest option is to calculate the partition function for each complex $c \in \Psi$ independently [11]. With this approach, as described previously, the partition function $Q_{1,N}$ for a complex with N nucleotides is calculated with a dynamic program that builds up from short subsequences to the full-length sequence, sweeping along each diagonal of the matrix of subsequence partition functions (Figure 2.4). This simplicity comes at the cost of some inefficiency, for when multiple copies of the same strand species appear in a complex, intermediate results appear in multiple locations within the matrix. Moreover, when the same strand species appears in multiple complexes, intermediate results appear in multiple matrices.

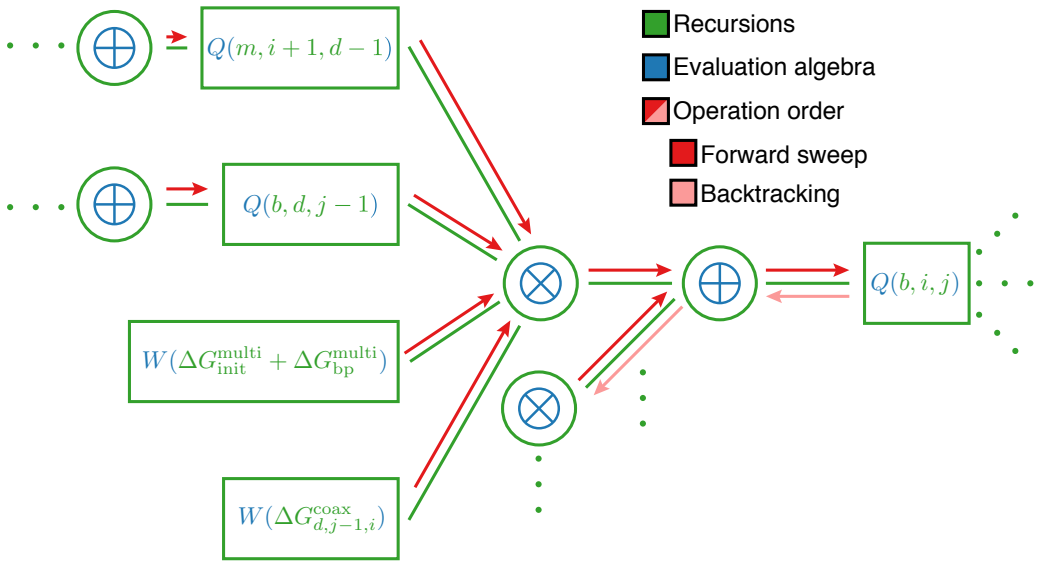


Figure 2.9: Conceptual interplay between three dynamic program ingredients: recursions, evaluation algebra, and operation order. Recursions specify the dependencies between subproblems and incorporate the details of the structural ensemble and free energy model. Evaluation algebras define the mathematical form of each subproblem. Operation orders specify the computational trajectory through the dependency graph of subproblems.

Here, we reduce the cost of calculating the partition functions for the set of complexes Ψ by decomposing each matrix into two types of subcomplex blocks (Figure 2.8a): triangular intrastrand blocks (e.g., blocks A, B, C) and rectangular interstrand blocks (e.g., blocks AB, BC, ABC). Blocks are computed in ascending order of the number of strands per block (blocks with the same number of strands can be calculated independently) and cached such that blocks arising in multiple locations within a complex or test tube ensemble are not recomputed (Figure 2.8b). Section S4.2 provides pseudocode for a blockwise operation order that is $O(N^3)$ for a complex of N nucleotides, including exact calculation of interior loop contributions [8, 18]. Moreover, with this blockwise operation order, recursions (Section S2) can be coded using vectorized dot products (Figure 2.8c) such that compilation with the appropriate evaluation algebra (Table 2.2) yields an efficient vectorized dynamic program for calculating the corresponding physical quantity. The interplay between the three dynamic programming ingredients (recursions, evaluation algebra, and operation order) is illustrated conceptually in Figure 2.9.

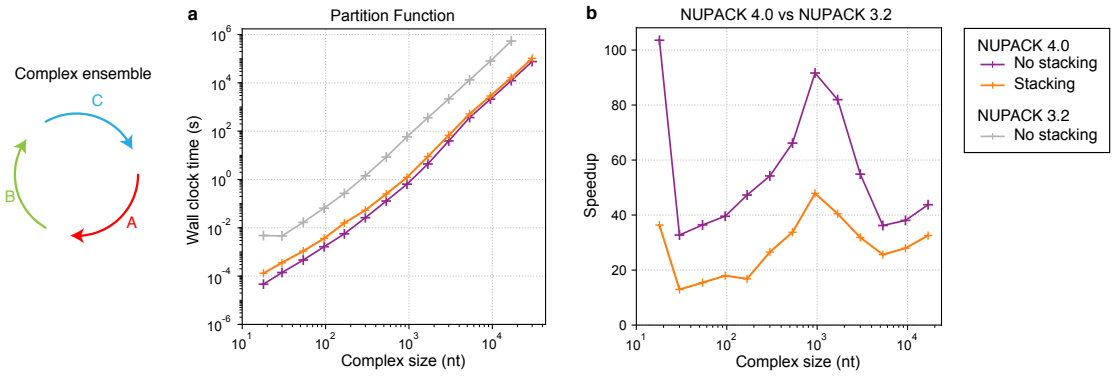


Figure 2.10: Enhanced efficiency for partition function calculations on complex ensembles including very large complexes. Calculation of the partition function for a complex of 3 strands, each with a different random sequence of uniform length. NUPACK 4.0 (vectorized, overflow-safe implementation, physical model with or without coaxial and dangle stacking) vs NUPACK 3.2 (not vectorized, quadruple-precision arithmetic, physical model with no coaxial or dangle stacking). (a) Computational cost. (b) Computational speedup (ratio of mean wall clock times). Means wall clock time over 5 sets of random sequences per complex size (due to overflow, results not available for largest complex size using NUPACK 3.2). Conditions: RNA, 37 °C, 1M Na^+.

2.2.8 Enhanced efficiency and scalability of the partition function algorithm for complex ensembles including very large complexes

Figure 2.10 highlights efficiency gains for partition function calculations on complex ensembles. Compared using the same physical model without coaxial and dangle stacking, the vectorized NUPACK 4.0 implementation yields ≈30–90× speedups over NUPACK 3.2 depending on the complex size. Operating on the enhanced physical model that includes coaxial and dangle stacking subensembles, NUPACK 4.0 continues to achieve speedups of ≈13–45× over NUPACK 3.2 operating on the simpler physical model that neglects these terms. Figure 2.11 demonstrates that the overflow-safe evaluation algebra SplitExp enables NUPACK 4.0 to calculate partition functions exceeding the overflow thresholds for single-, double-, and quadruple-precision floating point arithmetic. Note that the partition function grows faster as a function of complex size for designed sequences than for random sequences due to the presence of a deep well on designed free energy landscapes.

2.2.9 Enhanced efficiency of the partition function algorithm for sets of complexes in test tube ensembles

Figure 2.12 highlights efficiency gains for partition function calculations for sets of complexes in test tube ensembles. Blockwise caching yields an empirical speedup of $\approx(L_{\max}-1)$ for a range of test tube ensembles containing M strand species interacting to form all com-

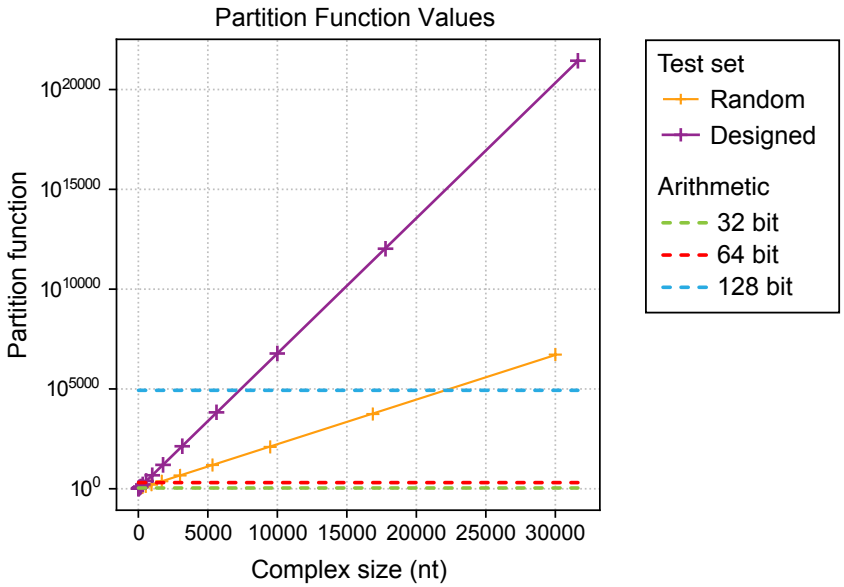


Figure 2.11: Overflow-safe partition function calculations on complex ensembles including very large complexes. Dashed lines denote the overflow thresholds for single-, double-, and quadruple-precision arithmetic. Partition function calculations performed using NUPACK 4.0 (overflow-safe implementation with coaxial and dangle stacking) for two test sets: random test set (complexes of 3 strands, each with a different random sequence of uniform length), designed test set (duplexes with designed sequences). Mean partition function over 5 sets of random or designed sequences per complex size. Conditions: RNA, 37 °C, 1M Na^+.

plexes of up to $L_{\max}$ strands (Figure 2.12a). Comparing the performance of NUPACK 4.0 (with the benefits of vectorization and blockwise caching but the added cost of an enhanced physical model with coaxial and dangle stacking) to NUPACK 3.2 (without these features) reveals speedups of ≈20× for test tubes containing all complexes of up to $L_{\max} = 2$ strands and up to ≈120× for test tubes containing all complexes up to $L_{\max} = 6$ strands. With NUPACK 4.0, Figure 2.13 illustrates the size of test tube ensembles for which equilibrium analysis can be performed in ≤ 1 minute on a single computational core (e.g., $M = 80$ strand species of 100 nt each interacting to form all complex species of up to $L_{\max} = 2$ strands, or $M = 2$ strand species of 300 nt each interacting to form all complex species of up to $L_{\max} = 6$ strands).

2.2.10 Backtrack-free base-pairing probability matrices

Historically, equilibrium base-pairing probabilities for a single strand [9, 21] or a complex [10] are calculated using a dynamic program that backtracks through the matrix of subsequence partition functions. This backtracking process involves subtraction of intermediate

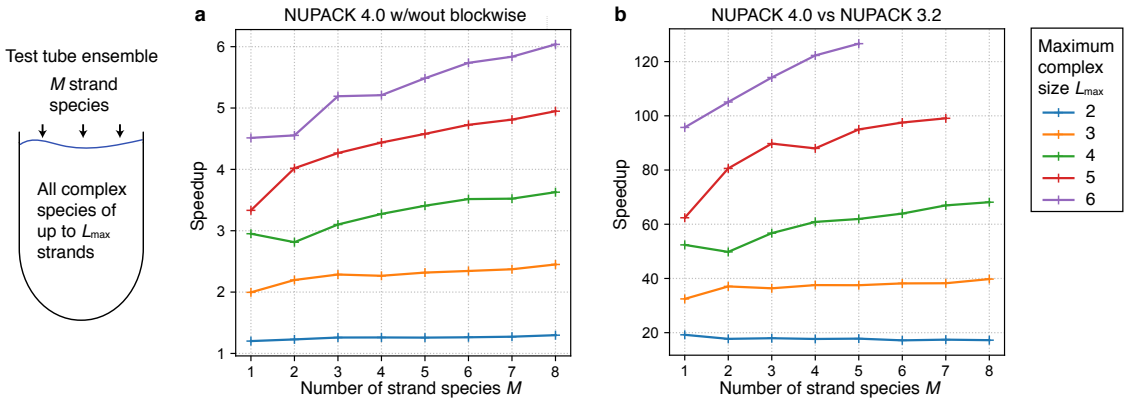


Figure 2.12: Enhanced efficiency of the partition function algorithm for sets of complexes in test tube ensembles. Calculation of the partition function for all complexes of up to $L_{\max}$ strands for a test tube ensemble containing M strand species, each with a different random 50 nt sequence. (a) Speedup with vs without blockwise caching for NUPACK 4.0. (b) Speedup using NUPACK 4.0 (vectorized, blockwise caching, enhanced physical model with coaxial and dangle stacking) vs NUPACK 3.2 (no blockwise caching, not vectorized, physical model with no coaxial or dangle stacking). Mean wall clock time over 10 sets of random sequences per test tube ensemble size. Conditions: RNA, 37 °C, 1M Na^+, each strand introduced at 10 nM.

partition function quantities, creating the risk of losing precision due to subtraction of large numbers differing by a small amount. To eliminate this concern, here we calculate equilibrium base-pairing probabilities without backtracking using the same blended evaluation algebras and a modification of the blockwise operation order that are used for overflow-safe partition function calculations.

To see how this is possible, consider a complex with strand ordering $\pi = \mathrm{ABC}$ and a total of N nucleotides. As an intermediate result, the partition function algorithm calculates $Q^b_{i,j}$, the conditional partition function for subsequence $i, \ldots, j$ subject to the constraint that i is paired to j. We may similarly calculate the conditional partition function, $Q^{b_{\text{ext}}}_{i,j}$, for the remaining nucleotides external to subsequence $i, \ldots, j$, namely nucleotides $j+1, \ldots, N, 1, \ldots, i-1$ (Figure 2.14a). Because the structural ensemble $\overline{\Gamma}(\phi)$ excludes pseudoknots, the base pair $i \cdot j$ partitions the structural ensemble into non-interacting internal and external ensembles, so the partition function of all structures containing base pair $i \cdot j$ is the product $Q^b_{i,j} Q^{b_{\text{ext}}}_{i,j}$. As a result, the equilibrium probability of base pair $i \cdot j$ over ensemble $\overline{\Gamma}(\phi)$ is given by

$$\overline{P}_{i,j}(\phi) = Q^b_{i,j}(\phi) Q^{b_{\text{ext}}}_{i,j}(\phi) / Q_{1,N}(\phi). \tag{2.16}$$

Mathews employed this approach using new recursions to calculate the external conditional partition function $Q^{b_{\text{ext}}}_{i,j}$ for a single strand[20]. Here, treating the general case of a complex of L strands, we observe that $Q^{b_{\text{ext}}}_{i,j}$ can be calculated in a straightforward way without new

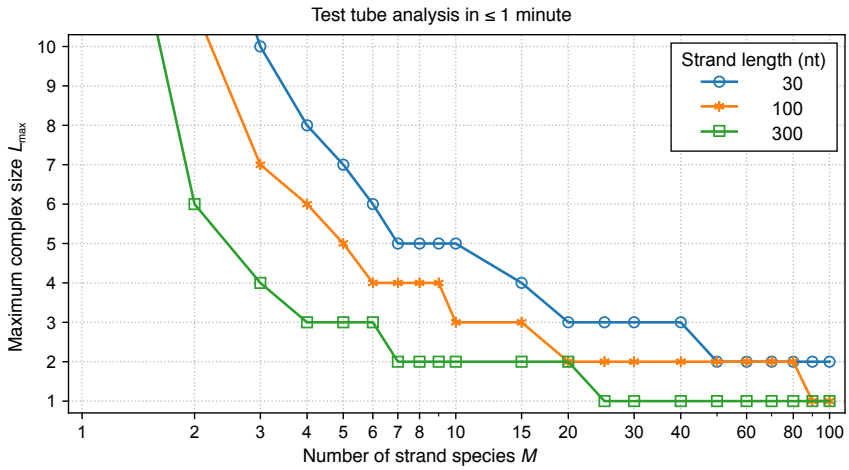


Figure 2.13: Equilibrium test tube analysis in under 1 minute. Calculation of the partition function and equilibrium complex concentration for a test tube ensemble containing M strand species that form all complexes of up to $L_{\max}$ strands. Symbols denote test tube ensembles for which the wall clock time $\leq$ 1 minute. After calculating the set of partition functions Q_Ψ for a given test tube ensemble Ψ, the set of equilibrium concentrations x_Ψ is obtained by solving the convex optimization problem (2.14). Mean wall clock time over 5 sets of random sequences per test tube ensemble size. Conditions: RNA, 37 °C, 1M Na^+, each strand introduced at 10 nM.

recursions by replicating the strands to form a "doubled" complex with sequence ϕ' (e.g., π = ABCABC) containing $2N$ nucleotides and calculating $Q^b_{i,j}$ using the standard recursions for all subsequences of up to N nucleotides (Figure 2.14b). The external subsequence $j+1,\ldots,N,1,\ldots,i-1$ for the original complex with sequence ϕ is simply the internal subsequence $j, N+i$ for the doubled complex with sequence ϕ'. Hence, we have:

$$\overline{P}_{i,j}(\phi) = Q^b_{i,j}(\phi) Q^b_{j,N+i}(\phi') / Q_{1,N}(\phi). \tag{2.17}$$

In Figure 2.14, the yellow blocks are previously cached from the partition function calculation. The orange entries correspond to calculation of $Q^b_{j,N+i}(\phi')$. The cost of evaluating each entry is proportional to subsequence length (the horizontal or vertical distance from the diagonal), so the average cost per entry in the orange block is higher than for the yellow blocks. Empirically, after calculating the partition function $\overline{Q}(\phi)$ at a cost C_Q, calculation of the equilibrium base-pairing probability matrix $\overline{P}(\phi)$ costs an additional $C_P \approx 1.5$–$3C_Q$ (Figure S40).

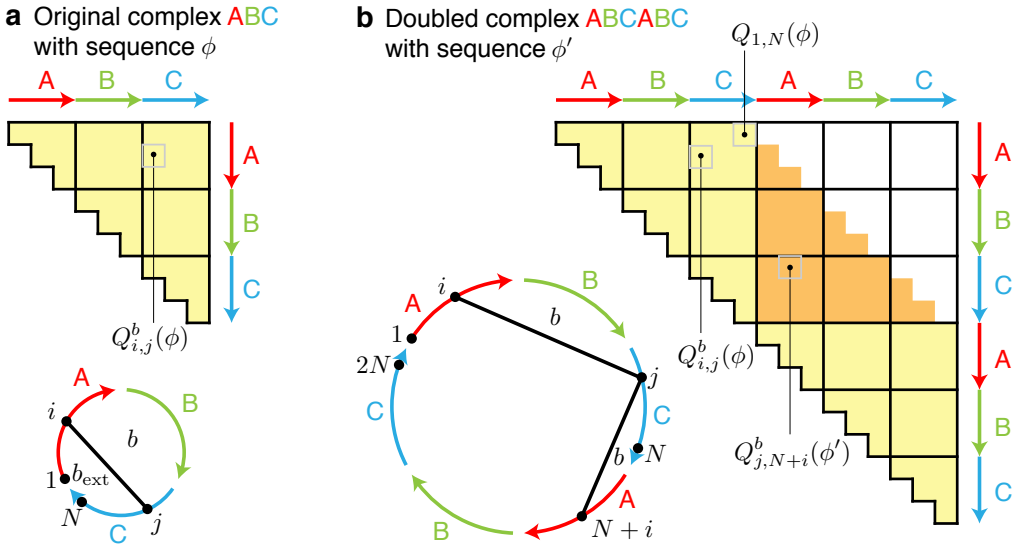


Figure 2.14: Backtrack-free calculation of the equilibrium base-pairing probability $P_{i,j}(\phi)$ for a complex ABC of N nucleotides with sequence ϕ using (2.17) and the conditional partition functions $Q^b_{i,j}(\phi)$ and $Q^b_{j,N+i}(\phi')$. The latter is calculated by considering the "doubled" complex ABCABC of $2N$ nucleotides with sequence ϕ'.

2.2.11 Evaluation algebras and backtracking operation orders for simultaneous structure sampling, MFE structure determination, and suboptimal structure determination

After calculating the partition function $\overline{Q}(\phi)$ for a strand [6] or a complex [11], a structure s_{sample} can be randomly sampled from the structural ensemble $\overline{\Gamma}(\phi)$ by backtracking through the matrix of subsequence partition functions. Likewise, after calculating the minimum free energy $\Delta G(\phi, s_{\text{MFE}})$ for a strand [36] or a complex [11], the corresponding MFE structure $s_{\text{MFE}}(\phi)$ can be determined by backtracking through the matrix of subsequence MFEs. These dynamic programs can be expressed in our unified dynamic programming framework (Figure 2.6) using the same set of recursion diagrams/equations (Section S2) as the forward algorithms, but employing new evaluation algebras (Table 2.2cd), and with the operation order reversed so the blockwise dependency tree (Figure 2.8b) is traversed top to bottom.

For structure sampling, backtracking starts from the recursion for $Q_{1,N}$ and for MFE structure determination, backtracking starts from the recursion for $F_{1,N}$. In either case, backtracking is used to "choose" between competing recursion elements when a $\oplus$ operator is encountered and to "join" compatible recursions elements when a $\otimes$ operator is encountered; the mathematical implementations of these operators are described by quantity-specific evaluation algebras (Table 2.2cd). For sampling, $\oplus$ corresponds to randomly choosing between com-

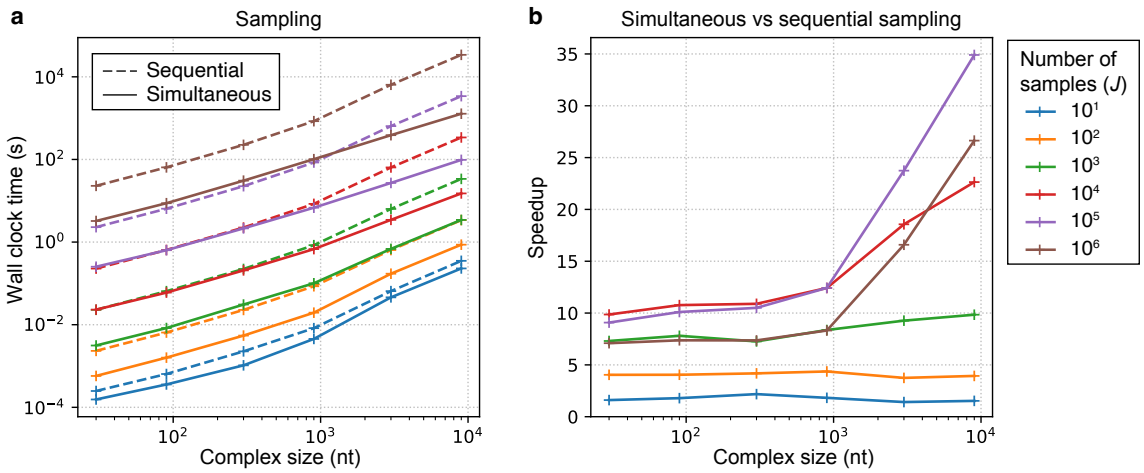


Figure 2.15: Enhanced efficiency for sampling multiple structures from complex ensembles using simultaneous rather than sequential sampling. Boltzmann sampling secondary structures for a complex of 3 strands, each with a different random sequence of uniform length. (a) Computational cost. (b) Computational speedup (ratio of mean wall clock times). Mean wall clock time over 10 sets of random sequences per complex size. Conditions: RNA, 37 °C, 1M Na^+. See Section S6.6 for additional data.

peting (Boltzmann-weighted) recursion elements, while for MFE structure determination, $\oplus$ corresponds to choosing the MFE of competing recursion elements. For both structure sampling and MFE structure determination, $\otimes$ corresponds to the set union $\cup$ of compatible recursion elements.

The MFE structure determination algorithm can be generalized to calculate the set of suboptimal structures $\overline{\Gamma}(\phi, \Delta G_{\text{gap}})$ within a specified free energy gap $\Delta G_{\text{gap}} \geq 0$ of the MFE using generalized evaluation operators for $\oplus$ and $\otimes$ (see Section S3.2.6). In practice, we implement this more general algorithm and then apply it with $\Delta G_{\text{gap}} = 0$ if the MFE structure proxy is requested. The number of suboptimal structures can grow rapidly with ΔG_{gap} and N so we perform backtracking using a stack data structure that reduces memory usage by generating complete structures at the earliest opportunity, enabling these structures to be emitted in a streaming fashion while additional structures are determined (see Section S4.6).

While the pair probability matrix $\overline{P}(\phi)$ provides the equilibrium probability of each base pair over the complex ensemble, it does not reveal correlation information between different base pairs. By sampling a set of J secondary structures and averaging or clustering over this set, it is possible to address questions like "what is the probability that a set of adjacent bases are simultaneously unpaired?"[6] or "is the free energy landscape dominated by multiple deep basins each defined by a set of related secondary structures?"[7]. Existing algorithms perform sequential sampling of J structures for a strand [6] ($O(JN^2)$ time complexity if

long interior loops are excluded) or a complex [11] ($O(JN^3)$ with exact treatment of interior loops). Motivated by the central use case where a set of J structures is needed for averaging or clustering, here we develop a simultaneous sampling approach that samples J structures all at once ($O(JN^2)$ with exact treatment of interior loops). A given recursion element may contribute to a large number of sampled structures (e.g., if there is a deep well on the free energy landscape), so we perform backtracking using a priority queue data structure that reduces computational effort by ensuring that all samples of any given recursion element are performed during a single visit to that recursion element (see Section S4.4). With the simultaneous sampling algorithm, we observe order-of-magnitude speedups over sequential sampling for J above $\approx 10^3$ (Figure 2.15), and empirical complexity $\sim J^{0.8}N^{1.2}$ for J samples from a random complex ensemble of N nucleotides (see Section S6.6).

2.3 Conclusions

The new unified dynamic programming framework combines recursions capturing the details of the physical model with quantity-specific evaluation algebras and operation orders to enable efficient and scalable calculation of diverse physical quantities over complex and test tube ensembles of interacting DNA or RNA strands. The physical model was upgraded by deriving recursions for the complex ensemble that include coaxial and dangle stacking subensembles for multiloops and exterior loops. The recursions are coded generically and then compiled with a quantity-specific evaluation algebra and operation order to generate an executable for each physical quantity. As a result, future upgrades to the physical model can be implemented by updating the generic recursions rather than by updating code for each physical quantity. For large complexes, scalability is achieved for partition function calculations using an overflow-safe evaluation algebra, and for equilibrium pair probabilities by using a backtrack-free operation order, enabling calculations on complexes containing 30,000 nt. For test tube ensembles, dramatic efficiency gains of 1–2 orders of magnitude are achieved using a new blockwise operation order that facilitates vectorization and caching. Recognizing that Boltzmann sampling is most useful for averaging or clustering information calculated on large set of structures, a new sampling algorithm yields order-of-magnitude speedups by sampling all requested structures simultaneously. These enhancements to the physical model, algorithm scalability, and algorithm speed represent substantial advances for researchers analyzing nucleic acid structures, devices, and systems. Moreover, these enhancements are directly applicable to sequence design algorithms operating over complex and test tube ensembles [30, 31, 34] as sequence analysis is the most costly component of

sequence design; work is underway to integrate these advances into the NUPACK 4.0 sequence design algorithms.

2.4 Methods summary

2.4.1 Implementation

NUPACK algorithms are programmed in the C++17 programming language. Dynamic programs are implemented using a generic programming paradigm [27] employing expression templates and compile-time polymorphism; generic recursion equations capturing the details of the structural ensemble and free energy model are translated via template metaprogramming into a separate vectorized executable for calculating each physical quantity in Table 2.2. Single-threaded single instruction multiple data (SIMD) vectorization is implemented using the Boost.SIMD library[12]. The convex optimization problem (2.14) is solved in the dual form using an efficient trust region method [11] using the Armadillo linear algebra library for matrix operations [23].

2.4.2 Trials

All benchmarks were run on AWS EC2 C5 instances (3.0 GHz Intel Xeon Platinum processors) with 72 GB of memory (except 144 GB for Figure 2.10).

2.5 Resources

2.5.1 NUPACK source code

The NUPACK source code can be downloaded for non-commercial academic use subject to the NUPACK License (`nupack.org`). NUPACK documentation includes a User Guide and example jobs.

2.5.2 NUPACK Python module

The all-new NUPACK Python interface allows streamlined and flexible in-memory scripting of NUPACK jobs, reducing file I/O and increasing the convenience of developing workflows composing multiple NUPACK commands.

BIBLIOGRAPHY

[1] Mirela Andronescu, Zhi Chuan Zhang, and Anne Condon. Secondary structure prediction of interacting RNA molecules. *Journal of Molecular Biology*, 345:987–1001, 2005.

[2] Stephan H. Bernhart, Hakim Tafer, Ulrike Mückstein, Christoph Flamm, Peter F. Stadler, and Ivo L. Hofacker. Partition function and base pairing probabilities of RNA heterodimers. *Algorithms for Molecular Biology*, 1(1):1–10, 2006.

[3] Victor Bloomfield and Donald M. Crothers. *Nucleic acids: Structures, properties and functions*. Number 574.192 B52. University Science Books, 2000.

[4] Salvatore Bommarito, Nicolas Peyret, and John SantaLucia. Thermodynamic parameters for DNA sequences with dangling ends. *Nucleic Acids Research*, 28(9): 1929–1934, 2000.

[5] Roumen A. Dimitrov and Michael Zuker. Prediction of hybridization and melting for double-stranded nucleic acids. *Biophysical Journal*, 87(1):215–226, 2004.

[6] Ye Ding and Charles E. Lawrence. A statistical sampling algorithm for RNA secondary structure prediction. *Nucleic Acids Research*, 31(24):7280–7301, 2003.

[7] Ye Ding, Chi Yu Chan, and Charles E. Lawrence. RNA secondary structure prediction by centroids in a Boltzmann weighted ensemble. *RNA*, 11(8):1157–1166, 2005.

[8] Robert M. Dirks and Niles A. Pierce. A partition function algorithm for nucleic acid secondary structure including pseudoknots. *Journal of Computational Chemistry*, 24 (13):1664–1677, 2003.

[9] Robert M. Dirks and Niles A. Pierce. An algorithm for computing nucleic acid base-pairing probabilities including pseudoknots. *Journal of Computational Chemistry*, 25 (10):1295–1304, 2004.

[10] Robert M. Dirks, Milo Lin, Erik Winfree, and Niles A. Pierce. Paradigms for computational nucleic acid design. *Nucleic Acids Research*, 32(4):1392–1403, 2004.

[11] Robert M. Dirks, Justin S. Bois, Joseph M. Schaeffer, Erik Winfree, and Niles A. Pierce. Thermodynamic analysis of interacting nucleic acid strands. *SIAM Review*, 49(1):65–88, 2007.

[12] Pierre Estérie, Joel Falcou, Mathias Gaunard, and Jean-Thierry Lapresté. Boost.SIMD: generic programming for portable SIMDization. In *Proceedings of the 2014 Workshop on Programming Models for SIMD/Vector Processing*, pages 1–8, 2014.

[13] Cody Geary, Paul W.K. Rothemund, and Ebbe S. Andersen. A single-stranded architecture for cotranscriptional folding of RNA nanostructures. *Science*, 345(6198): 799–804, 2014.

[14] Lisa M. Hochrein, Maayan Schwarzkopf, Mona Shahgholi, Peng Yin, and Niles A. Pierce. Conditional Dicer substrate formation via shape and sequence transduction with small conditional RNAs. *Journal of the American Chemical Society*, 135(46): 17322–17330, 2013. ISSN 00027863. doi: 10.1021/ja404676x.

[15] Ivo L. Hofacker, Walter Fontana, Peter F. Stadler, L. Sebastian Bonhoeffer, Manfred Tacker, and Peter Schuster. Fast folding and comparison of RNA secondary structures. *Monatshefte für Chemie/Chemical Monthly*, 125(2):167–188, 1994.

[16] Ryan T. Koehler and Nicolas Peyret. Thermodynamic properties of DNA sequences: Characteristic values for the human genome. *Bioinformatics*, 21(16):3333–3339, 2005.

[17] Zhi John Lu, Douglas H. Turner, and David H. Mathews. A set of nearest neighbor parameters for predicting the enthalpy change of RNA secondary structure formation. *Nucleic Acids Research*, 34(17):4912–4924, 2006.

[18] Rune B. Lyngsø, Michael Zuker, and C.N. Pedersen. Fast evaluation of internal loops in RNA secondary structure prediction. *Bioinformatics*, 15(6):440–445, 1999.

[19] David H. Mathews, Jeffrey Sabina, Michael Zuker, and Douglas H. Turner. Expanded sequence dependence of thermodynamic parameters improves prediction of RNA secondary structure. *Journal of Molecular Biology*, 288:911–940, 1999.

[20] 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.

[21] John S. McCaskill. The equilibrium partition function and base pair binding probabilities for RNA secondary structure. *Biopolymers: Original Research on Biomolecules*, 29(6-7):1105–1119, 1990.

[22] Nicolas Peyret. *Prediction of nucleic acid hybridization: Parameters and algorithms*. Thesis, Wayne State University, 2000.

[23] Conrad Sanderson and Ryan Curtin. Armadillo: a template-based C++ library for linear algebra. *Journal of Open Source Software*, 1(2):26, 2016.

[24] John SantaLucia. A unified view of polymer, dumbbell, and oligonucleotide DNA nearest-neighbor thermodynamics. *Proceedings of the National Academy of Sciences*, 95(4):1460–1465, 1998.

[25] John SantaLucia and Donald Hicks. The thermodynamics of DNA structural motifs. *Annual Review of Biophysics and Biomolecular Structure.*, 33:415–440, 2004.

[26] Martin J. Serra and Douglas H. Turner. Predicting thermodynamic properties of RNA. *Methods in Enzymology*, 259:242–261, 1995.

[27] Alexander A. Stepanov and Daniel E. Rose. *From mathematics to generic programming*. Pearson Education, 2014.

[28] Ignacio Tinoco, Olke C. Uhlenbeck, and Mark D. Levine. Estimation of secondary structure in ribonucleic acids. *Nature*, 230(5293):362–367, 1971.

[29] Douglas H. Turner and David H. Mathews. NNDB: The nearest neighbor parameter database for predicting stability of nucleic acid secondary structure. *Nucleic Acids Research*, 38(suppl_1):D280–D282, 2010.

[30] Brian R. Wolfe and Niles A. Pierce. Sequence design for a test tube of interacting nucleic acid strands. *ACS Synthetic Biology*, 4(10):1086–1100, 2015.

[31] Brian R. Wolfe, Nicholas J. Porubsky, Joseph N. Zadeh, Robert M. Dirks, and Niles A. Pierce. Constrained multistate sequence design for nucleic acid reaction pathway engineering. *Journal of the American Chemical Society*, 139(8):3134–3144, 2017.

[32] Tianbing Xia, John SantaLucia, Mark E. Burkard, Ryszard Kierzek, Susan J. Schroeder, Xiaoqi Jiao, Christopher Cox, and Douglas H. Turner. Thermodynamic parameters for an expanded nearest-neighbor model for formation of RNA duplexes with Watson-Crick base pairs. *Biochemistry*, 37(42):14719–14735, 1998.

[33] Joseph N. Zadeh, Conrad D. Steenberg, Justin S. Bois, Brian R. Wolfe, Marshall B. Pierce, Asif R. Khan, Robert M. Dirks, and Niles A. Pierce. NUPACK: Analysis and design of nucleic acid systems. *Journal of Computational Chemistry*, 32(1):170–173, 2011. doi: 10.1002/jcc.21596.

[34] Joseph N. Zadeh, Brian R. Wolfe, and Niles A. Pierce. Nucleic acid sequence design via efficient ensemble defect optimization. *Journal of Computational Chemistry*, 32 (3):439–452, 2011.

[35] Michael Zuker. Mfold web server for nucleic acid folding and hybridization prediction. *Nucleic Acids Research*, 31(13):3406–3415, 2003.

[36] Michael Zuker and Patrick Stiegler. Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. *Nucleic Acids Research*, 9(1):133–148, 1981.

Chapter 3

EFFICIENT SIMULATION OF NUCLEIC ACID SECONDARY STRUCTURE STOCHASTIC KINETIC TRAJECTORIES

This chapter is temporarily embargoed.

Chapter 4

MOLECULAR DYNAMICS METHODS FOR SIMULATING NUCLEIC ACID BASE-PAIRING REACTIONS

This chapter is temporarily embargoed.

Chapter 5

COMPUTATIONAL PARAMETERIZATION OF EQUILIBRIUM AND KINETIC NUCLEIC ACID SECONDARY STRUCTURE MODELS

This chapter is temporarily embargoed.

Appendix A

ADDITIONAL DETAILS FOR IMPROVED ALGORITHMS FOR THE EQUILIBRIUM ANALYSIS OF NUCLEIC ACID COMPLEXES

S1 Additional free energy model details

S1.1 Strand association penalty for a complex

Based on dimensional analysis, we define the complex concentrations x_Ψ for a test tube containing the set of complexes Ψ as mole fractions rather than molarities (see (2.14)). Therefore, we adjust the strand association penalty

$$\Delta G^{\text{assoc}} = \Delta G^{\text{assoc}}_{\text{pub}} - kT \log[\rho_{H_2O}/(1 \text{ mol/liter})] \tag{S1}$$

where $\Delta G^{\text{assoc}}_{\text{pub}}$ is the published value for two strands associating [21] and ρ_{H_2O} is the molarity of water (e.g., $\rho_{H_2O} = 55.14$ mol/liter at 37 °C).[7] The strand association penalty for a complex of L strands (see (2.1)) is then

$$(L-1)\Delta G^{\text{assoc}}. \tag{S2}$$

S1.2 Salt corrections for DNA complexes

The default salt conditions for RNA [10, 14, 20, 22, 24, 26] and DNA [2, 17, 18, 26] parameter sets are [NaCl] = 1 M. Salt corrections are available for DNA parameters [9, 17–19] to permit calculations in user-specified sodium, potassium, ammonium, and magnesium ion concentrations. Following SantaLucia and co-workers, the free energy of a DNA duplex at 37°C is augmented by

$$-0.114 \frac{N}{2} \log[Na^+], \tag{S3}$$

for user-specified $0.05\,\text{M} \le [Na^+] \le 1.0\,\text{M}$, where N is the number of phosphates in the duplex and it is assumed that ΔH is independent of $[Na^+]$, which is valid for this salt regime [18, 19]. This salt correction was derived using duplexes with 16 bp or less and the accuracy decreases as duplex length increases further [18, 19]. The expression can be generalized to monovalent potassium and ammonium ions [19] as well as to divalent magnesium cations[9, 17]:

$$-0.114 \frac{N}{2} \log\left([Na^+] + [K^+] + [NH_4^+] + 3.3\,[Mg^{++}]^{1/2}\right), \tag{S4}$$

for user-specified for $0.05\,\mathrm{M} \leq [\mathrm{Na^+}] + [\mathrm{K^+}] + [\mathrm{NH_4^+}] \leq 1.0\,\mathrm{M}$ and $0.0\,\mathrm{M} \leq [\mathrm{Mg^{++}}] \leq 0.2\,\mathrm{M}$.

To apply this salt correction to a complex of L strands at temperature T, consider a secondary structure s containing one or more duplexes. We assume that strands are synthesized with one phosphate per base so that $N/2 = n_{\mathrm{bp}}(s)$ where N is the total number of phosphates in duplexes and $n_{\mathrm{bp}}(s)$ is the total number of base pairs in s. (If strands are synthesized without a $5'$ terminal phosphate, then N approximates the total number of phosphates in duplexes.) We further assume that ΔH is independent of cation concentration in this regime. The secondary structure free energy $\Delta G(\phi, s)$ is then augmented by

$$n_{\mathrm{bp}}(s)\Delta G^{\mathrm{salt}} \tag{S5}$$

with

$$\Delta G^{\mathrm{salt}} = -0.114\,\log\left([\mathrm{Na^+}] + [\mathrm{K^+}] + [\mathrm{NH_4^+}] + 3.3\,[\mathrm{Mg^{++}}]^{1/2}\right)\frac{T}{T_{37}} \tag{S6}$$

for user-specified

$$0.05\,\mathrm{M} \leq [\mathrm{Na^+}] + [\mathrm{K^+}] + [\mathrm{NH_4^+}] \leq 1.0\,\mathrm{M}, \tag{S7}$$

$$0.0\,\mathrm{M} \leq [\mathrm{Mg^{++}}] \leq 0.2\,\mathrm{M}, \tag{S8}$$

with $T_{37} = 310.15$ K. In order to incorporate this salt correction in dynamic programs without explicitly calculating $n_{\mathrm{bp}}(s)$, note that for a complex of L strands, the total number of loops in each secondary structure is

$$n_{\mathrm{loop}}(s) = n_{\mathrm{bp}}(s) + 1. \tag{S9}$$

This may be seen, for example, by starting with a single strand with no base pairs (corresponding to a single exterior loop). Each addition of a base pair adds one loop. Once all base pairs in s have been added, each addition of a nick increases the number of strands by one without changing the number of loops (all secondary structures in the complex ensemble are connected so introduction of each nick converts a loop from another type to an exterior loop). Let $n_{\mathrm{loop}}^{\mathrm{other}}$ denote the total number of non-exterior loops and $n_{\mathrm{loop}}^{\mathrm{exterior}}$ denote the total number of exterior loops, so we have

$$n_{\mathrm{loop}}(s) = n_{\mathrm{loop}}^{\mathrm{exterior}}(s) + n_{\mathrm{loop}}^{\mathrm{other}}(s). \tag{S10}$$

For a complex of L strands, $n_{\mathrm{loop}}^{\mathrm{exterior}}(s) = L$. Thus, the salt correction (S5) becomes

$$n_{\mathrm{bp}}(s)\Delta G^{\mathrm{salt}} = (L-1)\Delta G^{\mathrm{salt}} + n_{\mathrm{loop}}^{\mathrm{other}}(s)\Delta G^{\mathrm{salt}}. \tag{S11}$$

Hence, the salt correction can be implemented by adding

$$\Delta G^{\text{salt}} \tag{S12}$$

to every $\Delta G(\text{loop})$ except for exterior loops as a pre-processing step, using our suite of dynamic programs without modification, and then treating the constant term $(L-1)\Delta G^{\text{salt}}$ in a post-processing step (see Section S1.4).

S1.3 Temperature dependence

The loop-based free energy model (2.1) is temperature dependent. Each loop free energy is calculated using

$$\Delta G(\text{loop}) = \Delta H(\text{loop}) - T\Delta S(\text{loop}) \tag{S13}$$

where T is in Kelvin and $\Delta H(\text{loop})$ and $\Delta S(\text{loop})$ are assumed to be temperature independent [19]. Model parameters are provided for RNA [10, 14, 20, 22, 24, 26] and DNA [2, 17, 18, 26] in the form of $\Delta G_{37}(\text{loop})$ and $\Delta H(\text{loop})$ which can be used to calculate

$$\Delta S(\text{loop}) = \frac{1}{T_{37}}[\Delta H(\text{loop}) - \Delta G_{37}(\text{loop})] \tag{S14}$$

with $T_{37} = 310.15$ K, so (S13) becomes

$$\Delta G(\text{loop}) = \Delta H(\text{loop}) - \frac{T}{T_{37}}[\Delta H(\text{loop}) - \Delta G_{37}(\text{loop})]. \tag{S15}$$

Similarly, for the strand association penalty (S1):

$$\Delta G^{\text{assoc}}_{\text{pub}} = \Delta H^{\text{assoc}}_{\text{pub}} - T\Delta S^{\text{assoc}}_{\text{pub}}. \tag{S16}$$

and the provided parameters $\Delta G^{\text{assoc}}_{37,\text{pub}}$ and $\Delta H^{\text{assoc}}_{\text{pub}}$ yield

$$\Delta G^{\text{assoc}}_{\text{pub}} = \Delta H^{\text{assoc}}_{\text{pub}} - \frac{T}{T_{37}}[\Delta H^{\text{assoc}}_{\text{pub}} - \Delta G^{\text{assoc}}_{37,\text{pub}}]. \tag{S17}$$

The temperature dependence is explicit in the form of the symmetry correction (2.5) and salt correction (S5).

S1.4 Treatment of constant free energy terms for complex ensembles

Consider a complex of L strands containing a total of N nucleotides. Suppose that free energy model terms have been pre-processed as described above for units [see (S1)], salt

corrections [see (S12)], and temperature corrections [see (S15 and S16)] prior to calculating any physical quantities. The secondary structure free energy (2.1) then becomes

$$\overline{\Delta G}(\phi, s) = (L-1)[\Delta G^{\text{assoc}} + \Delta G^{\text{salt}}] + \sum_{\text{loop}\in s} \Delta G(\text{loop}). \tag{S18}$$

After running the partition function dynamic program to calculate $Q_{1,N}$, the partition function is then

$$\overline{Q}(\phi) = \exp\{-(L-1)[\Delta G^{\text{assoc}} + \Delta G^{\text{salt}}]/kT\} Q_{1,N}. \tag{S19}$$

where this post-processing step accounts for the constant terms ΔG^{assoc} and ΔG^{salt} that affect all secondary structures in the complex ensemble. Likewise, after running the MFE dynamic program to calculate $F_{1,N}$, the free energy of the MFE stacking state is then

$$\overline{\Delta G}(\phi, s^{||}_{\text{MFE}}) = (L-1)[\Delta G^{\text{salt}} + \Delta G^{\text{assoc}}] + F_{1,N}. \tag{S20}$$

The equilibrium base-pairing probability $\overline{P}_{i,j}$ is calculated via (2.17) using the values of $Q^b_{i,j}(\phi)$, $Q^b_{j,N+i}(\phi')$ and $Q_{1,N}(\phi)$ returned by the dynamic program; the constant terms ΔG^{assoc} and ΔG^{salt} do not affect the calculation as they are omitted in both the numerator and the denominator of (2.17). The dynamic programs for calculating the MFE proxy structure, suboptimal structures, or sampled structures are unaffected by the constant terms ΔG^{assoc} and ΔG^{salt} so no post-processing is required for those quantities.

S1.5 RNA and DNA parameter sets

NUPACK 4 algorithms perform calculations on the following complex ensembles:

- `stacking`: with coaxial and dangle stacking (ensemble $\overline{\Gamma}^{||}(\phi)$).
- `nostacking`: without coaxial and dangle stacking (ensemble $\overline{\Gamma}(\phi)$).

These ensembles can be used for calculations in combination with the following temperature-dependent DNA and RNA parameter sets:

- `rna95` based on Serra and Turner [20] with additional parameters [26] including coaxial stacking [14, 22] and dangle stacking [20, 22, 26] in 1M Na^+.
- `dna04` based on SantaLucia [18] and SantaLucia and Hicks [19] with additional parameters [26] including coaxial stacking [17] and dangle stacking [2, 26] in user-specified concentrations of Na^+, K^+, NH_4^+, and Mg^{++} (see Section S1.2 for details on implementation of the salt corrections) [9, 17–19].

- `rna06` based on Mathews et al. [14], Mathews et al. [15], and Lu et al. [10] with additional parameters [24, 26] including coaxial stacking [14, 22] and dangle stacking [20, 22, 26] in 1M Na^+.
- `custom` based on user-specified parameters representing nucleic acids or synthetic nucleic acid analogs in experimental conditions of choice.

Base pairs are either Watson-Crick pairs (`G·C` and `A·U` for RNA; `G·C` and `A·T` for DNA) or wobble pairs (`G·U` for RNA). Note that for DNA, `G` and `T` form a mismatch and not a wobble pair [19].

S1.6 Historical RNA and DNA parameter sets (for backwards compatibility with NUPACK 3)

For backwards compatibility, the following historical complex ensembles without coaxial stacking and with approximate dangle stacking are supported (see Section S2.5):

- `none-nupack3`: no dangle stacking and no coaxial stacking (dangles "`none`" option for NUPACK 3)
- `some-nupack3`: some dangle stacking and no coaxial stacking (dangles "`some`" option for NUPACK 3)
- `all-nupack3`: all dangle stacking and no coaxial stacking (dangles "`all`" option for NUPACK 3)

For these historical ensembles, base pairs are either Watson-Crick pairs (`G·C` and `A·U` for RNA; `G·C` and `A·T` for DNA) or wobble pairs (`G·U` for RNA; `G·T` for DNA). Note that for the historical ensembles, `G·T` is classified as a DNA wobble pair and not as a mismatch. The historical ensembles prohibit a wobble pair (`G·U` or `G·T`) as a terminal base pair in an exterior loop or a multiloop. As a result, an attempt to evaluate a free energy for a sequence ϕ and secondary structure s that place a wobble pair as a terminal base pair in an exterior loop or multiloop will return $\overline{\Delta G}(\phi, s) = \Delta G(\phi, s) = \infty$. These historical ensembles can be used for calculations in combination with the following historical DNA and RNA parameter sets:

- `rna95-nupack3` is the same as `rna95` except that terminal mismatch free energies in exterior loops and multiloops are replaced by two dangle stacking free energies (see equation (S55)).

- `dna04-nupack3` is the same as `dna04` except that `G·T` was treated as a wobble pair (analogous to a `G·U` RNA wobble pair) instead of classifying `G` and `T` as a mismatch. Note that while terminal mismatch free energies in exterior loops and multiloops are replaced by two dangle stacking free energies (see equation (S55)), this is the same treatment as in `dna04`, as terminal mismatch parameters are not public for DNA [19].

- `rna99-nupack3` based on Mathews et al. [14] with additional parameters [24, 26] including dangle stacking [20, 22, 26] in 1M Na^+. Terminal mismatch free energies in exterior loops and multiloops are replaced by two dangle stacking free energies (see equation (S55)). Parameters are provided only for 37 °C.

S1.7 Functional form of RNA and DNA free energy models

S1.7.1 Free energy model for hairpin loops

A hairpin loop is defined for a subsequence $\phi_{[i:j]}$ by the single base pair $i \cdot j$ such that there are no nicks or additional base pairs in the range $[i : j]$. Let $n \equiv j - i - 1$ denote the number of unpaired nucleotides in the hairpin loop. Steric effects are assumed to prevent hairpin loops with $n < 3$ for both RNA [20, 22] and DNA [19]. The functional form of the hairpin free energy is as follows:

$$\Delta G^{\text{hairpin}}(\phi_{[i:j]}) = \Delta G^{\text{hairpin}}_{\text{size}}(n) + \Delta G^{\text{hairpin}}_{\text{seq}}(\phi_{[i:j]}) \tag{S21}$$

For the size-dependent term [10, 14, 19, 20]:

$$\Delta G^{\text{hairpin}}_{\text{size}}(n) = \begin{cases} \infty, & n < 3 \\ \Delta G^{\text{hairpinsize}}_{n}, & 3 \le n \le 30 \\ \Delta G^{\text{hairpinsize}}_{30} + \log\left(\frac{n}{30}\right) \Delta G^{\text{polymer}}_{\text{entropy}}, & n > 30 \end{cases} \tag{S22}$$

- $\Delta G^{\text{hairpinsize}}_{n}$: a lookup table up to $n = 30$. `rna95` and `rna06` populate the lookup table using empirical values of $\Delta G^{\text{hairpinsize}}_{n}$ up to $n = 9$ and logarithmic extrapolation for larger n [10, 14, 20]. `dna04` populates the lookup table using empirical values of $\Delta G^{\text{hairpinsize}}_{n}$ for a subset of $3 \le n \le 30$ and logarithmic interpolation for the other values [19].
- $\Delta G^{\text{polymer}}_{\text{entropy}}$: a logarithmic extrapolation parameter based on Jacobson-Stockmayer polymer theory for $n > 30$. `rna95`, `dna04`, and `rna06` use previously published values [19, 20].

For the sequence-dependent term:[10, 14, 19]

$$\Delta G^{\text{hairpin}}_{\text{seq}}(\phi_{[i:j]}) = \begin{cases} \Delta G^{\text{triloop}}_{\phi_{[i:j]}} + \Delta G^{\text{terminalbp}}_{\phi_j,\phi_i} & n = 3 \\ \Delta G^{\text{tetraloop}}_{\phi_{[i:j]}} & n = 4 \\ \Delta G^{\text{hairpinmm}}_{\phi_{j-1},\phi_j,\phi_i,\phi_{i+1}} & n \ge 5 \end{cases} \tag{S23}$$

- $\Delta G^{\text{triloop}}_{\phi_{[i:j]}}$: sequence-dependent penalty for hairpin loop of length $n = 3$. 0 kcal/mol for `rna95` [20]. Empirical values for `dna04` [19] and `rna06` [15].
- $\Delta G^{\text{terminalbp}}_{\phi_i,\phi_j}$: sequence-dependent penalty for non-GC terminal base pair at the end of a duplex. Empirical values for `rna95` [20], `dna04` [19], and `rna06` [15].

- $\Delta G^{\text{tetraloop}}_{\phi_{[i:j]}}$: sequence-dependent penalty for hairpin loop of length $n = 4$. Empirical values for `rna95` [20], `dna04` [19], and `rna06` [15].

- $\Delta G^{\text{hairpinmm}}_{\phi_{j-1},\phi_j,\phi_i,\phi_{i+1}}$: sequence-dependent term for mismatched bases adjacent to base pair $i \cdot j$. Empirical values set equal to $\Delta G^{\text{terminalmm}}_{\phi_{j-1},\phi_j,\phi_i,\phi_{i+1}}$ plus sequence-dependent modifications for `rna95` [20] and `rna06` [10, 14]. Empirical values for $\Delta G^{\text{terminalmm}}_{\phi_{j-1},\phi_j,\phi_i,\phi_{i+1}}$ not public for DNA [19], so $\Delta G^{\text{hairpinmm}}_{\phi_{j-1},\phi_j,\phi_i,\phi_{i+1}}$ set to unpublished values made available in the Mfold software [26] for `dna04`. (See multiloops and exterior loops for a description of $\Delta G^{\text{terminalmm}}_{\phi_{j-1},\phi_j,\phi_i,\phi_{i+1}}$).

S1.7.2 Free energy model for interior loops

An interior loop may be defined via a pair of subsequences $\phi_{[i:d]}$ and $\phi_{[e:j]}$ such that $i < d < e < j$ with base pairs $i \cdot j$ and $d \cdot e$, with no additional paired bases or nicks within the two subsequences.

Stacked pairs. Stacked pairs are the special case where $d = i + 1$ and $j = e + 1$.

$$\Delta G^{\text{stackedpair}}(\phi_{[i:i+1]}, \phi_{[j-1:j]}) = \Delta G^{\text{stack}}_{\phi_j,\phi_i,\phi_{i+1},\phi_{j-1}} \tag{S24}$$

- $\Delta G^{\text{stack}}_{\phi_j,\phi_i,\phi_{i+1},\phi_{j-1}}$: the stack free energy has been determined for all allowable base pair combinations from experimental results for `rna95` [20], `rna06` [10, 14, 24], and `dna04` [19].

Bulge loops. A bulge loop is the special case with either $d = i + 1$ or $j = e + 1$ but not both. Here, we will outline the functional form when $d = i + 1$. Let $n \equiv j - e - 1$ denote the number of unpaired nucleotides in the bulge loop.

$$\Delta G^{\text{bulge}}(\phi_{[i:i+1]}, \phi_{[e:j]}) = \Delta G^{\text{bulge}}_{\text{size}}(n) + \Delta G^{\text{bulge}}_{\text{seq}}(\phi_{[i:i+1]}, \phi_{[e:j]}). \tag{S25}$$

For the size-dependent term:

$$\Delta G^{\text{bulge}}_{\text{size}}(n) = \begin{cases} \Delta G^{\text{bulgesize}}_{n}, & n \leq 30 \\ \Delta G^{\text{bulgesize}}_{30} + \log\left(\frac{n}{30}\right) \Delta G^{\text{polymer}}_{\text{entropy}}, & n > 30 \end{cases} \tag{S26}$$

- $\Delta G_n^{\text{bulge size}}$: `rna95` uses empirical values for $1 \le n \le 5$ [20]. `rna06` uses empirical values for $1 \le n \le 6$ [10, 14]. `dna04` uses empirical values for a subset of $1 \le n \le 30$ [19]. Each parameter set uses a logarithmic approximation for all other values of n.

For the sequence-dependent term:

$$\Delta G_{\text{seq}}^{\text{bulge}}(\phi_{[i:i+1]}, \phi_{[e:j]}) = \begin{cases} \Delta G_{\phi_j,\phi_i,\phi_{i+1},\phi_e}^{\text{stack}}, & e+2=j \\ \Delta G_{\phi_j,\phi_i}^{\text{terminalbp}} + \Delta G_{\phi_{i+1},\phi_e}^{\text{terminalbp}}, & \text{otherwise.} \end{cases} \tag{S27}$$

Other small interior loops. The free energies for interior loops with $2 \le d - i \le 3$ and $2 \le j - e \le 3$ are kept in a lookup table.

- **1×1 interior loop.** Corresponds to $d-i=2$ and $j-e=2$. `rna95` assigns a sequence-independent ΔG [20]. `rna06` uses unpublished parameters made available in the Mfold software [26]. `dna04` models these loops using (S28) below [19]; a positive constant free energy is assigned for mismatches where the unpaired nucleotides are Watson-Crick complements [26].

- **1 × 2 interior loop.** Corresponds to $d - i = 2$ and $j - e = 3$, or $d - i = 3$ and $j - e = 2$. `rna95` and `dna04` model these loops using (S28) below [19, 20]. For `dna04`, a positive constant free energy is assigned for mismatches where the unpaired nucleotides are Watson-Crick complements [26]. `rna06` models these loops using a combination of tabulated data and averaging [10, 14].

- **2 × 2 interior loop.** Corresponds to $d - i = 3$ and $j - e = 3$. `rna95` and `dna04` model these loops using (S28) below [19, 20]. For `dna04`, a positive constant free energy is assigned for mismatches where the unpaired nucleotides are Watson-Crick complements [26]. `rna06` models these loops using tabulated symmetric tandem interior mismatches and averaging for asymmetric tandem interior mismatches [10, 14].

Other interior loops. Let $n_1 \equiv d-i-1$ and $n_2 \equiv j-e-1$ denote the number of unpaired nucleotides for the two sides of the interior loop. For the general case of interior loops not handled via special cases above, the following formula is used:

$$\Delta G^{\text{interior}}(\phi_{[i:d]}, \phi_{[e:j]}) = \Delta G^{\text{interior}}_{\text{size}}(n_1 + n_2) + \Delta G^{\text{interior}}_{\text{asymm}}(n_1, n_2) + \Delta G^{\text{interior}}_{\text{mm}}(\phi_{[i:d]}, \phi_{[e:j]}). \tag{S28}$$

For the size-dependent term:

$$\Delta G^{\text{interior}}_{\text{size}}(n) = \begin{cases} \Delta G^{\text{interiorsize}}_{n}, & n \leq 30 \\ \Delta G^{\text{interiorsize}}_{30} + \log\left(\frac{n}{30}\right) \Delta G^{\text{polymer}}_{\text{entropy}}, & n > 30 \end{cases} \tag{S29}$$

- $\Delta G^{\text{interiorsize}}_{n}$: `rna95` uses empirical values for $2 \leq n \leq 6$ [20]. `rna06` uses empirical values for $4 \leq n \leq 6$ [10, 14]. `dna04` uses empirical values for a subset of values in $3 \leq n \leq 30$ [19]. Each parameter set uses a logarithmic approximation for all other values of n.

For the asymmetry-based term:

$$\Delta G^{\text{interior}}_{\text{asymm}}(n_1, n_2) = \min(\Delta G^{\text{interiorasymm}}_{4}, |n_1 - n_2| \Delta G^{\text{interiorasymm}}_{\min(4,n_2,n_1)}) \tag{S30}$$

- $\Delta G^{\text{interiorasymm}}_{n}$: `rna95` [20], `rna06` [10, 14], and `dna04` [19] use values regressed from empirical data.

For the mismatch-based term:

$$\Delta G^{\text{interior}}_{\text{mm}}(\phi_{[i:d]}, \phi_{[e:j]}) = \begin{cases} \Delta G^{\text{interiormm}'}_{\phi_{j-1},\phi_j,\phi_i,\phi_{i+1}} + \Delta G^{\text{interiormm}'}_{\phi_{d-1},\phi_d,\phi_e,\phi_{e+1}} & i+2 = d \text{ or } e+2 = j \\ \Delta G^{\text{interiormm}}_{\phi_{j-1},\phi_j,\phi_i,\phi_{i+1}} + \Delta G^{\text{interiormm}}_{\phi_{d-1},\phi_d,\phi_e,\phi_{e+1}} & \text{otherwise} \end{cases} \tag{S31}$$

- $\Delta G^{\text{interiormm}}_{\phi_{j-1},\phi_j,\phi_i,\phi_{i+1}}$: `rna95` [20] and `rna06` [10, 14] use independently determined values for loops without complementary unpaired bases. `dna04` equates $\Delta G^{\text{interiormm}}_{\phi_{j-1},\phi_j,\phi_i,\phi_{i+1}}$ with $\Delta G^{\text{terminalmm}}_{\phi_{j-1},\phi_j,\phi_i,\phi_{i+1}}$ [19], which are assigned unpublished values made available in the Mfold software [26].

- $\Delta G^{\text{interiormm}'}_{\phi_{j-1},\phi_j,\phi_i,\phi_{i+1}}$: `rna95` [20], `rna06` [10, 14], `dna04` [19] use different parameters for the case when one side of the interior loop has only one unpaired nucleotide [15].

S1.7.3 Free energy model for multiloops

A multiloop contains 3 or more terminal base pairs and no nicks. It may be defined as a series of bounding subsequences $[\phi]$. If the number of terminal base pairs is n_{bp} and the number of unpaired nucleotides is n_{nt}, the free energy for a multiloop in a specified stacking state, ω, is modeled as follows:

$$\begin{aligned}\Delta G^{\text{multi}}([\phi],\omega) = \Delta G^{\text{multi}}_{\text{init}} + n_{\text{bp}}\Delta G^{\text{multi}}_{\text{bp}} + n_{\text{nt}}\Delta G^{\text{multi}}_{\text{nt}} \\ + \Delta G^{\text{allterminalbp}}([\phi]) + \Delta G^{\text{allcoax}}([\phi],\omega) + \Delta G^{\text{alldangle}}([\phi],\omega)\end{aligned} \quad \text{(S32)}$$

where $\Delta G^{\text{multi}}_{\text{init}}$ denotes the penalty for formation of a multiloop, $\Delta G^{\text{multi}}_{\text{bp}}$ denotes the sequence-independent penalty for a terminal base pair in a multiloop, and $\Delta G^{\text{multi}}_{\text{nt}}$ denotes the penalty per unpaired nucleotide in a multiloop. Note that in contrast to interior loops and hairpin loops, the free energy of a multiloop is assumed to scale linearly, not logarithmically, with the number of unpaired nucleotides; the linear simplification facilitates the derivation of $O(N^3)$ multiloop recursions.

- $\Delta G^{\text{multi}}_{\text{init}}$: empirical values for `rna95` [20]; newly regressed values (Table A.1) for `rna06` [13]; unpublished values for `dna04` [26].
- $\Delta G^{\text{multi}}_{\text{bp}}$: empirical values for `rna95` [20]; newly regressed values (Table A.1) for `rna06` [13]; unpublished values for `dna04` [26].
- $\Delta G^{\text{multi}}_{\text{nt}}$: empirical values for `rna95` [20]; newly regressed values (Table A.1) for `rna06` [13]; unpublished values for `dna04` [26].

Note that for `rna06`, previously published parameter regressions [10, 14] use a functional form incompatible with the definition of $\Delta G^{\text{multi}}([\phi],\omega)$ above [20, 26]. Using literature source data for multiloops [13], we regressed the values of the $\Delta G^{\text{multi}}_{\text{init}}$, $\Delta G^{\text{multi}}_{\text{bp}}$, $\Delta G^{\text{multi}}_{\text{nt}}$ via a least-squares fit of the regressed loop free energies, observing comparable mean absolute error (Table A.1).

$\Delta G^{\text{allterminalbp}}([\phi])$ is a sum of the sequence-dependent free energy $\Delta G^{\text{terminalbp}}_{\phi_i,\phi_j}$ for each terminal base pair $i \cdot j$ in the multiloop (see definition above under hairpin loops).

$\Delta G^{\text{allcoax}}([\phi],\omega)$ is a sum over each coaxial stack present in the multiloop stacking state ω. Owing to a lack of parameters, only coaxial stacks between adjacent terminal base

pairs (with no intervening unpaired bases) are considered. Each coaxial stack between base pairs $i \cdot d$ and $d+1 \cdot j$ contributes a free energy of $\Delta G^{\text{coax}}_{\phi_i,\phi_d,\phi_{d+1},\phi_j}$. For the recursions with coaxial stacking (Section S2.6), we use $\Delta G^{\text{coax}}_{i,d,j}(\phi)$ to denote $\Delta G^{\text{coax}}_{\phi_i,\phi_d,\phi_{d+1},\phi_j}$ since only three indices may vary freely. For the recursions without coaxial stacking (Section S2.3), the term $\Delta G^{\text{allcoax}}([\phi],\omega)$ is neglected.

- $\Delta G^{\text{coax}}_{\phi_i,\phi_d,\phi_{d+1},\phi_j}$: `rna95` [20] and `rna06` [10, 14] set $\Delta G^{\text{coax}}_{\phi_i,\phi_d,\phi_{d+1},\phi_j}$ equal to $\Delta G^{\text{stack}}_{\phi_i,\phi_d,\phi_{d+1},\phi_j}$. `dna04` uses independently estimated values [17].

$\Delta G^{\text{alldangle}}([\phi],\omega)$ is a sum of the sequence-dependent free energy, $\Delta G^{\text{dangle}}_{i,j}(\phi)$, for each terminal base pair $i \cdot j$ that is not in a coaxial stack in stacking state ω. For a given terminal base pair $i \cdot j$, $\Delta G^{\text{dangle}}_{i,j}(\phi)$ takes one of four values to match the dangle stacking state for a given ω:

$$\Delta G^{\text{dangle}}_{i,j}(\phi) = \begin{cases} 0 & \text{no dangles} \\ \Delta G^{5'\text{dangle}}_{\phi_i,\phi_{i+1},\phi_j} & 5' \text{ dangle} \\ \Delta G^{3'\text{dangle}}_{\phi_i,\phi_{j-1},\phi_j} & 3' \text{ dangle} \\ \Delta G^{\text{terminalmm}}_{\phi_i,\phi_{i+1},\phi_{j-1},\phi_j} & \text{terminal mismatch} \end{cases} \tag{S33}$$

Note that the state where both 5′ and 3′ dangles stack on terminal base pair $i \cdot j$ is classified as a terminal mismatch. For the recursions without dangle stacking (Section S2.3), the term $\Delta G^{\text{alldangle}}([\phi],\omega)$ is neglected.

- $\Delta G^{5'\text{dangle}}_{\phi_i,\phi_j,\phi_k}$: 5′ dangle free energy parametrized for `rna95` [20], `rna06` [10, 14], and `dna04` [19].
- $\Delta G^{3'\text{dangle}}_{\phi_i,\phi_j,\phi_k}$: 3′ dangle free energy parametrized for `rna95` [20], `rna06` [10, 14], and `dna04` [19].

Quantity	$\Delta[G/H]^{\text{init}}_{\text{multi}}$	$\Delta[G/H]^{\text{bp}}_{\text{multi}}$	$\Delta[G/H]^{\text{nt}}_{\text{multi}}$	MAE	MAE[13]
ΔG	+12.91	-1.28	-0.0880	1.01	1.01
ΔH	+81.06	-6.84	+2.22	11.5	12.1

Table A.1: Regression of multiloop parameters for `rna06` (kcal/mol). MAE denotes the mean absolute error of the least-squares regression of the loop free energies from Reference [13] using formulation (S32) for $\Delta G^{\text{multi}}([\phi],\omega)$. MAE [13] refers to the mean absolute error of the regression performed in Reference [13] using a different formulation of $\Delta G^{\text{multi}}([\phi],\omega)$.

- $\Delta G^{\text{terminalmm}}_{\phi_i,\phi_{i+1},\phi_{j-1},\phi_j}$: `rna95` [20] and `rna06` [10, 14] use empirical parameters for $\Delta G^{\text{terminalmm}}$; `dna04` assigns $\Delta G^{\text{terminalmm}}_{\phi_i,\phi_{i+1},\phi_{j-1},\phi_j}$ to be the sum of $\Delta G^{5'\text{dangle}}_{\phi_i,\phi_{i+1},\phi_j}$ and $\Delta G^{3'\text{dangle}}_{\phi_i,\phi_{j-1},\phi_j}$ as empirical values of $\Delta G^{\text{terminalmm}}_{\phi_i,\phi_{i+1},\phi_{j-1},\phi_j}$ are not publicly available [19].

S1.7.4 Free energy model for exterior loops

An exterior loop is a loop containing one nick and zero or more terminal base pairs. An exterior loop may be defined as a series of bounding subsequences $[\phi]$ with a given nick location. An unpaired strand is an exterior loop with a free energy of zero, corresponding to the reference state [7]. The free energy of an exterior loop in a specified stacking state ω is modeled as follows:

$$\Delta G^{\text{exterior}}([\phi],\omega) = 0+\Delta G^{\text{allterminalbp}}([\phi])+\Delta G^{\text{allcoax}}([\phi],\omega)+\Delta G^{\text{alldangle}}([\phi],\omega). \quad \text{(S34)}$$

The functions $\Delta G^{\text{allterminalbp}}([\phi])$, $\Delta G^{\text{allcoax}}([\phi],\omega)$, and $\Delta G^{\text{alldangle}}([\phi],\omega)$ are defined as above for multiloops. For the recursions without coaxial and dangle stacking (Section S2.3), the terms $\Delta G^{\text{allcoax}}([\phi],\omega)$ and $\Delta G^{\text{alldangle}}([\phi],\omega)$ are neglected.

S2 Recursions for the complex ensemble with or without coaxial and dangle stacking

Recursions specify the dependencies between subproblems and incorporate the details of the complex structural ensemble and the free energy model. The recursions described here can be combined with a quantity-specific evaluation algebra (Section S3) and a quantity-specific operation order (Section S4) to calculate diverse physical quantities. Each recursion corresponds to an efficient iteration through a *conditional ensemble* of substructures within a given subsequence that are compatible with a specified set of constraints. For a given recursion, a conditional ensemble might include an explicit structural element, which can be considered the base case of the recursion, or a reference to the result of another recursion.

S2.1 Separate recursions for intrastrand and interstrand blocks

Reference [7] described dynamic programming recursions for the complex ensemble that checked for a nick next to each nucleotide. This approach enabled treatment of complexes containing an arbitrary number of strands, but caused unnecessary complications in the program flow and eliminated any possibility of vectorization due to the conditional checks within each "for" loop. By contrast, here we employ separate sets of recursions for triangular intrastrand blocks and rectangular interstrand blocks (Figure S1). As a result, each intrastrand and interstrand recursion is kept as simple as possible and both types of recursions can be efficiently vectorized.

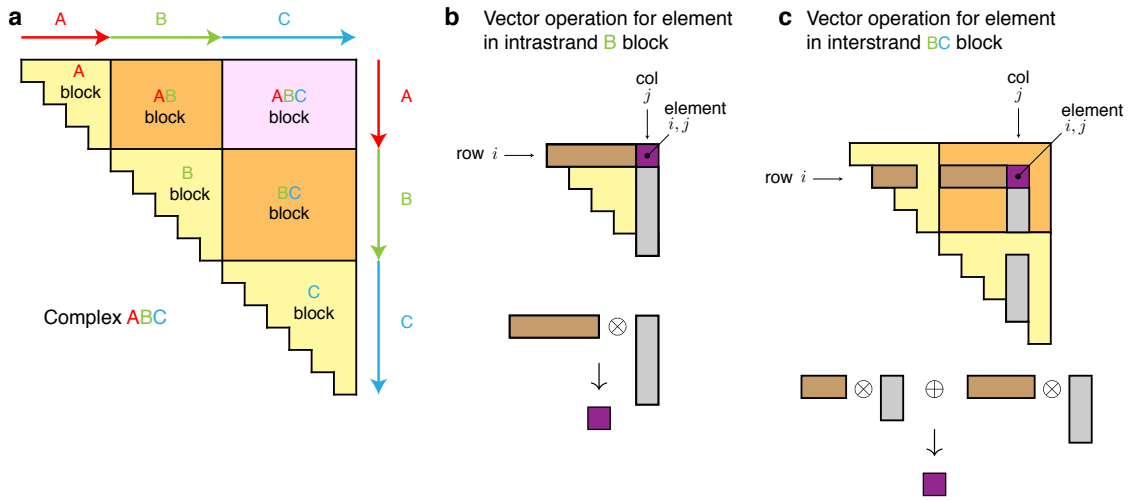


Figure S1: Separate recursions for intrastrand and interstrand blocks. (a) Triangular intrastrand blocks (A, B, C) and rectangular interstrand blocks (AB, BC, ABC) for complex ABC. Element i, j corresponds to a conditional ensemble for subsequence $[i, j]$ which contains no nicks if i, j is in an intrastrand block and one or more nicks if i, j is in an interstrand block. (b) Each recursion operation for calculation of element i, j in an intrastrand block (e.g., $Q_{i,j} \leftarrow \sum_{i \le d < j} Q_{i,d} Q_{d+1,j}$) can be implemented as a vectorized dot product between a subvector of row i (brown) and a subvector of column j (gray) to obtain element i, j (purple). Note that calculation of an element i, j in an intrastrand block uses elements in the same intrastrand block (calculated using intrastrand recursions). (c) Each recursion operation for calculation of element i, j in an interstrand block (e.g., $Q_{i,j} \leftarrow \sum_{i \le d < j,\ \mathrm{strand}(d)=\mathrm{strand}(d+1)} Q_{i,d} Q_{d+1,j}$) can be implemented as multiple vectorized dot products between valid subvectors of row i (brown) and valid subvectors of column j (gray) to obtain element i, j (purple), where valid positions are those that avoid introducing disconnected structures into the complex ensemble (see Algorithm S2). Note that calculation of element i, j in an interstrand block uses elements in one or more interstrand blocks (calculated with interstrand recursions) and two intrastrand blocks (calculated with intrastrand recursions).

S2.2 Conventions for recursion diagrams and equations

In the following sections we will describe recursions corresponding to the complex ensemble without stacking terms (Section S2.3) and with coaxial and dangle stacking subensembles (Section S2.6). Each recursion iterates over all conditional ensembles compatible with the constraints defined for a given recursion type. For a complex of N nucleotides, each full set of recursions is $O(N^3)$ in time and $O(N^2)$ in space. For interior loop recursions, we

start by defining an $O(N^4)$ recursion and then describe an exact reduction to $O(N^3)$ time complexity (Section S2.4).

Each recursion is represented in two ways: graphically, as a set of recursion diagrams, and algebraically, as an equation defining the recursion as a specific combination of contributions. The recursion diagrams employ the following conventions:

- Solid circular arcs depict the nucleic acid backbone. An arrowhead denotes the 3′ end of a strand.
- Dots indicate particular nucleotide positions that define the bounds of recursive contributions. If a dot is labeled with a nucleotide index, the same index is used in the corresponding recursion. If a dot is adjacent to a dot labeled i, the implied index of the unlabeled dot is either $i - 1$ or $i + 1$ (indices increase from 5′ to 3′).
- A straight line delimits the boundary for a given contribution. A solid straight line indicates that the connected nucleotides are base-paired. A dashed straight line indicate that the connected nucleotides may or may not be base-paired. A half-solid/half-dashed straight line indicates that the nucleotide connected on the solid side is base-paired to a nucleotide within the demarcated region. A straight line that is solid at both ends and dashed in the middle indicates that the nucleotides at either end are both base-paired but not to each other. A dotted straight line indicates that the connected nucleotides are involved in a stacking state (either a coaxial stacking state or a dangle stacking state).
- Shading indicates that the shaded region in a recursion explicitly incorporates a *recursion energy*, ΔG, representing all or part of a loop free energy (e.g., multiloop recursion energies representing different terms in the multiloop model are incorporated in multiple places in multiple recursions in order to treat the full multiloop model). The color of the shading corresponds to the loop type (and the stacking type when applicable).

A recursion equation provides a mathematical description of the conditional ensemble depicted graphically in a recursion diagram. Recursion equations employ the following conventions:

- For each physical quantity, an appropriate evaluation algebra (Section S3) is used to define the generic operators that appear in the recursion equations: $\mathbb{0}$, $\mathbb{1}$, $\oplus$, $\otimes$, W,

and Q. For example, to calculate the partition function, we have:

$$\mathbb{0} \to 0, \quad \mathbb{1} \to 1, \quad \oplus \to +, \quad \otimes \to \times, \quad W(g) \to \exp(-g/kT), \quad Q^a_{i,j} \to Q^a_{i,j}. \tag{S35}$$

where the last right-hand side indicates that $Q^a_{i,j}$ is a lookup of the relevant stored matrix element.

- A recursion equation for subsequence $[i : j]$ corresponding to element i, j in a triangular intrastrand block is denoted $R^a_{\text{INTRA}}(i, j, \phi)$ for a recursion of type a (e.g., $a \in \{\varnothing, b, m, \dots\}$). A recursion equation for subsequence $[i : j]$ corresponding to element i, j in a rectangular interstrand block is denoted $R^a_{\text{INTER}}(i, j, \phi)$ for a recursion of type a. Here, ϕ is the sequence of the complex and i and j are nucleotide indices. Note that in the Supporting Information we use $Q^{\varnothing}_{i,j}$ to denote $Q_{i,j}$ so that each recursion has an explicit recursion type a.

- If a recursion diagram contains a shaded region denoting a recursion energy, ΔG, the corresponding recursion equation will incorporate the recursion energy via the term $W(\Delta G)$.

- After it is evaluated for the first time, $R^a(i, j, \phi)$ is used to yield $Q^a_{i,j}$ in subsequent recursions. In the evaluation algebras that generate scalars (SumProduct, MinSum, Count), the output of $R^a(i, j, \phi)$ is synonymous with the value $Q^a_{i,j}$ that is stored in the recursion matrices. However, other evaluation algebras involve different treatment of the output of $R^a(i, j, \phi)$. For instance, a recursion in the SplitExp evaluation algebra (Section S3.1.4) yields a function that must be supplied with a reference exponent γ to calculate the mantissa and exponent values that are stored. The ways in which recursion outputs are utilized for each physical quantity are described in Section S4.

In our pseudocode, we make clear which operations are vectorized using SIMD operations on contiguous arrays via the function dot (Algorithm S1), which represents a dot product generalized to any number of arguments, each of which is a vector of the same length n. The **vectors** argument to this subroutine is composed of row or column subvectors (each a vector of contiguous elements) of the recursion matrices storing the result of previous recursion evaluations (e.g., $Q^{\varnothing}, Q^b, Q^m, \dots$). To denote a vector extracted from a matrix block, we replace a scalar index (e.g., d) with a vector index (e.g., $\overline{d}$) representing a range of either row or column indices. For example:

$$\overline{d} \equiv [i : j - 5] \equiv i, i + 1, \dots, j - 6, j - 5. \tag{S36}$$

```
DOT(vectors)
1  n ← LENGTH(vectors_1)
2  x ← 𝕆
3  for i ∈ [1 : n]
4       t ← 𝟙
5       for a ∈ vectors
6            t ← t ⊗ a_i
7       x ← x ⊕ t
8  return x
```

Algorithm S1: Generalized dot product over multiple vectors of equal length.

represents an ascending range of indices. Any scalar increment is applied to each entry in the range:

$$\overline{d} + 1 \equiv [i + 1 : j - 4]. \tag{S37}$$

$Q^{\varnothing}_{i,\overline{d}}$ then denotes a subvector of row i from matrix $Q^{\varnothing}$, $Q^{s}_{\overline{d}+1,j}$ denotes a subvector of column j from matrix Q^{s}, and

$$\textsc{dot}\left(Q^{\varnothing}_{i,\overline{d}}, Q^{s}_{\overline{d}+1,j}\right) \tag{S38}$$

denotes a dot product between these two vectors. An index range can also be used to denote a vector of free energy contributions, for example,

$$\overline{n}_{\text{nt}} \Delta G^{\text{multi}}_{\text{nt}} \tag{S39}$$

with

$$\overline{n}_{\text{nt}} \equiv [0 : j - i - 4]. \tag{S40}$$

When there are multiple ranges, the elements match with each other such that

$$a + b_{[i:j]} + c_{[d:e]} \equiv a + b_i + c_d, a + b_{i+1} + c_{d+1}, \ldots, a + b_{j-1} + c_{e-1}, a + b_j + c_e. \tag{S41}$$

In some cases, two ranges must proceed in opposite directions (one ascending and one descending) to match up the values in vectors correctly. A descending, or *reversed*, range is written

$$[i : j]^r \equiv j, j - 1, \ldots, i + 1, i. \tag{S42}$$

```
VALID(i, j, η)
 1  D ← {}
 2  m ← FIRST(η)
 3  n ← LAST(η)
 4  if i + 1 < m and j ≥ n
 5        d̄ = [i : m − 2]
 6        D ← D ∪ d̄
 7  if i < m and j − 1 ≥ n
 8        d̄ = [n : j − 1]
 9        D ← D ∪ d̄
10  if i < m and j ≥ n
11        for b ∈ [1 : LENGTH(η) − 1]
12              if η_{b+1} − η_b > 1
13                    d̄ = [η_b : η_{b+1} − 2]
14                    D ← D ∪ d̄
15  return D
```

Algorithm S2: Enumerate valid positions for vectorization in an interstrand block. η is an array of indices of the nicks between strands within the interstrand block being considered; by convention, each nick is denoted in η by the index of the nucleotide following the nick. The algorithm identifies at most one valid range $\overline{d}$ for each strand in the block, corresponding to the values of the index d such that d and $d + 1$ are on the same strand. This requirement ensures that all secondary structures are connected and that exterior loops only appear when they are being explicitly considered by a recursion. The algorithm returns **D**, the set of valid ranges for the block. For a given i and j, each valid range leads to a dot product between range $\overline{d}$ of row i and range $\overline{d} + 1$ of column j (e.g., the recursion of Figure S8 contains one dot product for each valid range $\overline{d}$).

For calculation of matrix elements in interstrand blocks (which by definition involve 2 or more strands), η is an array of indices of the nicks between strands within the interstrand block being considered; by convention, each nick is denoted in η by the index of the nucleotide following the nick.* For example, consider complex ABC of Figure S1a with strands A, B, and C containing 4, 5, and 6 nucleotides, respectively. For the AB block, $\eta = [5]$. For the BC block, $\eta = [10]$. For the ABC block, $\eta = [5, 10]$.

To calculate matrix entry i, j for an interstrand block with nicks η, the function VALID(i, j, η) (Algorithm S2) returns the set of valid ranges $\{\overline{d}_1, \overline{d}_2, \dots\}$ for vectorization so as to ensure that all secondary structures are connected and exterior loops appear only when they are explicitly considered by a recursion. There is at most one valid vectorization range per

*Note that this definition is unrelated to the use of η in Reference [7] to denote the number of nicks in a given subsequence.

strand, and there may be none for a strand that is too short or for the first or last strand if i or j, respectively, is too close to a nick. The ability to identify valid vectorization ranges for calculating each matrix element is a key innovation enabled by using dedicated recursions for intrastrand and interstrand blocks, eliminating the use "if" statements to identify nick locations (cf. Reference [7]), and thus enabling vectorization to achieve dramatic speedups.

Steric requirements require that there be at least three intervening bases between two base-paired nucleotides on the same strand, placing a lower bound on the length of subsequence $[i, j]$ for different recursion types (e.g., a minimum subsequence length to contain a hairpin loop, an interior loop, a multiloop, a terminal base pair, a stacking state, a coaxial stacking state, or a dangle stacking state). Recursions below the minimum subsequence length for a given recursion type return $\mathbb{0}$. For efficiency reasons, we often explicitly specify lower bounds on subsequence length to avoid performing calculations for elements that will evaluate to $\mathbb{0}$.

For exterior loop and multiloop recursions without coaxial and dangle stacking, the elementary recursion entity is a *terminal base pair* (a base pair that terminates a duplex to form a part of the exterior loop or multiloop). For exterior and multiloop recursions with coaxial and dangle stacking, the elementary recursion entity is the *stacking state*, representing either a *coaxial stacking state* (two adjacent terminal base pairs that are coaxially stacked) or a *dangle stacking state* (zero, one, or two unpaired nucleotides dangle stacking on an adjacent terminal base pair).

S2.3 Recursions without coaxial and dangle stacking subensembles

Here, we describe $R^a_{\text{INTRA}}(i, j, \phi)$ recursions for calculating the elements of intrastrand blocks and $R^a_{\text{INTER}}(i, j, \phi)$ recursions for calculating the elements of interstrand blocks for the complex ensemble, $\overline{\Gamma}$, without coaxial and dangle stacking subensembles. To assist with examining these recursions, the intuition behind the name chosen for each recursion, the nature of the ensemble treated by each recursion, and the dependencies between the different recursions is summarized in Figure S2. For convenience, it may be helpful to consider the recursions from the perspective of partition function calculations since there is a natural correspondence between the generic evaluation algebra nomenclature and the specific operators needed for partition function calculations (see equation (S35)), but the recursions are generic and can be combined with a quantity-specific evaluation algebra (Section S3) and a quantity-specific operation order (Section S4) to calculate diverse physical quantities. The present recursions treat the same structural ensemble $\overline{\Gamma}$ and free energy model $\overline{\Delta G}(\phi, s)$ as our previous implementation (NUPACK 3.2 with dangles option

"none")[7]. For backwards compatibility, we have also implemented the "some" and "all" approximate dangle treatments supported by NUPACK 3.2 (see Section S2.5).

A recursion $R^a(i, j, \phi)$ operates on subsequence $[i : j]$ to calculate element i, j for either the unconstrained ensemble $a = \varnothing$ or for one of several constrained ensembles $a \in \{s, b, x, ms, m\}$. Briefly, $R^{\varnothing}(i, j, \phi)$ treats the unconstrained ensemble in an exterior loop context where i and j may or may not be paired. $R^s(i, j, \phi)$ serves as an efficiency wrapper over the 3′-most terminal base pair in an exterior loop context to reduce the time complexity from $O(N^4)$ to $O(N^3)$. $R^b(i, j, \phi)$ treats the constrained ensemble where i and j form base pair $i \cdot j$ in the context of any loop type. $R^x(i, j, \phi)$ treats extensible interior loops to reduce the time complexity from $O(N^4)$ to $O(N^3)$. $R^{ms}(i, j, \phi)$ serves as an efficiency wrapper over the 3′-most terminal base pair in a multiloop context (analogous to R^s in an exterior loop context) to reduce the time complexity from $O(N^4)$ to $O(N^3)$. $R^m(i, j, \phi)$ treats the remaining terminal base pairs in a multiloop context.

Recursion	Naming intuition	Constraint	Context
$\varnothing$	unconstrained	none	exterior loop
s	**s**ummation	efficiency wrapper of 3′-most terminal base pair	exterior loop
b	**b**ase-paired	base pair between 5′-most and 3′-most bases of subsequence	any loop
x	e**x**tensible	extensible interior loop	interior loop
ms	**m**ultiloop **s**ummation	efficiency wrapper of 3′-most terminal base pair	multiloop
m	**m**ultiloop	one or more remaining terminal base pairs	multiloop

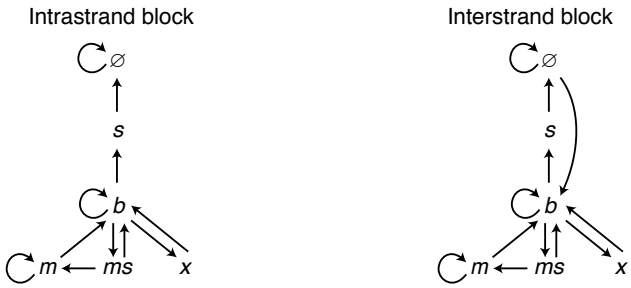


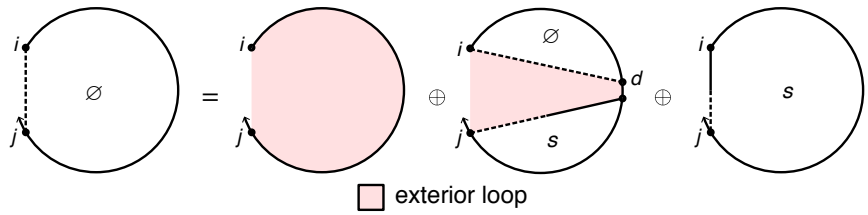
Figure S2: Nomenclature and connectivity for recursions without coaxial and dangle stacking. Top: Nomenclature. Bottom: Dependencies between different recursion types for elements within an intrastrand block (left) or an interstrand block (right).

S2.3.1 Intrastrand dynamic programming recursions without coaxial and dangle stacking

Here, we consider recursions for calculating the entries in a triangular intrastrand block without coaxial and dangle stacking. By definition, there are no nicks between strands in intrastrand recursions since intrastrand blocks involve base-pairing within a single strand.

$R^{\varnothing}_{\text{INTRA}}$ recursion without coaxial and dangle stacking. We begin with the recursion $R^{\varnothing}_{\text{INTRA}}(i, j, \phi)$ with the diagram and equation shown in Figure S3. $R^{\varnothing}_{\text{INTRA}}(i, j, \phi)$ operates on the unconstrained ensemble for subsequence $[i, j]$ in an exterior loop context where i and j may or may not be paired (depicted with a dashed line between i and j in the recursion diagram). This recursion distinguishes two cases that are combined using $\oplus$ in the recursion equation:

- *No terminal base pairs:* the empty case in an exterior loop context where there are no terminal base pairs in subsequence $[i, j]$ (depicted by the absence of a straight solid line in the recursion diagram). The shading in the recursion diagram represents the recursion energy $\Delta G^{\text{exterior}}_{i,j}(\phi) = 0$ corresponding to the zero reference state for an exterior loop with no base pairs and no coaxial or dangle stacking. The corresponding contribution to the recursion equation is $W(0) = \mathbb{1}$.



$$R^{\varnothing}_{\text{INTRA}}(i, j, \phi) \equiv \mathbb{1} \oplus \begin{cases} Q^{s}_{i,j} \oplus \text{DOT}\left(Q^{\varnothing}_{i,\overline{d}}, Q^{s}_{\overline{d}+1,j}\right), & j - i > 4 \\ Q^{s}_{i,j}, & j - i = 4 \\ \mathbb{0}, & \text{otherwise} \end{cases}$$

where $\overline{d} \equiv [i : j - 5]$.

Figure S3: $R^{\varnothing}_{\text{INTRA}}$ recursion without coaxial and dangle stacking. Top: recursion diagram. Bottom: recursion equation.

- *At least one terminal base pair:* the non-empty case in an exterior loop context where there is at least one terminal base pair (i.e., a base pair terminating a duplex) in subsequence $[i, j]$. The 3′-most terminal base pair begins at $d + 1$ and ends in the interval $[d + 2, j]$ (depicted using a half-solid/half-dashed line in the recursion diagram). The contributions for subsequence $[d + 1, j]$ are incorporated using a $Q^s_{d+1,j}$ element. Contributions for the remaining subsequence $[i, d]$ are incorporated by a $Q^{\varnothing}_{i,d}$ element. The shading denotes the recursion energy 0 corresponding to the zero reference state in an exterior loop context. Note that the recursion energy $\Delta G^{\text{terminalbp}}(\phi)$ representing one component of the $\Delta G^{\text{exterior}}_{i,j}(\phi)$ free energy is not incorporated here because the full identity of the terminal base pair (i.e., a base pair terminating a duplex) beginning at $d + 1$ is not known within the $R^{\varnothing}_{\text{INTRA}}(i, j, \phi)$ recursion (only within the $R^s_{\text{INTRA}}(i, j, \phi)$ recursion). The edge case where the index $d+1 = i$ is displayed explicitly to indicate that no $Q^{\varnothing}$ element is accessed in this case. The index limits in the recursion equation reflect the fact that steric effects prevent a hairpin loop with fewer than 3 unpaired nucleotides (hence, $i \cdot j$ cannot form if $j - i < 4$).

Note that using the DOT notation (Algorithm S1) and index range notation (S36) to denote vector operations, we have the equivalence:

$$\text{DOT}\left(Q^{\varnothing}_{i,\overline{d}}, Q^s_{\overline{d}+1,j}\right) \equiv \sum_{d=i}^{j-5} Q^{\varnothing}_{i,d} \otimes Q^s_{d+1,j}, \quad j - i > 4,$$

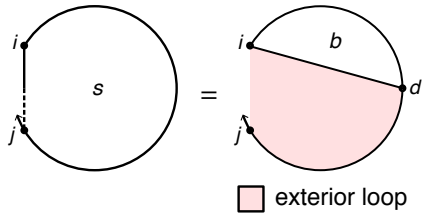
$$\text{where } \overline{d} \equiv [i : j - 5].$$

We can also recognize that in terms of matrix elements, the dot product

$$\text{DOT}\left(Q^{\varnothing}_{i,\overline{d}}, Q^s_{\overline{d}+1,j}\right) \tag{S43}$$

is between the element range $\overline{d}$ of row i (depicted as brown elements in Figure S1b) and the element range $\overline{d}$+1 of column j (gray elements), yielding element i, j (purple element).

R^s_{INTRA} recursion without coaxial and dangle stacking. The $R^{\varnothing}_{\text{INTRA}}(i, j, \phi)$ recursion references Q^s elements that are computed using the R^s_{INTRA} recursion displayed in Figure S4. $R^s_{\text{INTRA}}(i, j, \phi)$ operates on a conditional ensemble for subsequence $[i, j]$ in an exterior loop context containing one terminal base pair starting at i and ending in the interval $[i + 1, j]$ (depicted as a half-solid/half-dashed line between i and j). The contribution for the subsequence $[i, d]$ enclosed by base pair $i \cdot d$ is incorporated using a $Q^b_{i,d}$ element.



$$R^s_{\text{INTRA}}(i,j,\phi) \equiv \begin{cases} \text{DOT}\left(Q^b_{i,\overline{d}}, W(\Delta G^{\text{terminalbp}}_{i,\overline{d}}(\phi))\right), & j-i \geq 4 \\ \mathbb{0}, & \text{otherwise} \end{cases}$$

where $\overline{d} \equiv [i+4:j]$.

Figure S4: R^s_{INTRA} recursion without coaxial and dangle stacking. Top: recursion diagram. Bottom: recursion equation.

Shading corresponds to the recursion energy, $\Delta G^{\text{terminalbp}}_{i,d}(\phi)$, representing the sequence-dependent penalty for a terminal base pair in an exterior loop context (dependent on the sequence of base pair $i \cdot d$). The index limits in the recursion equation reflect the fact that steric effects prevent a hairpin loop with fewer than 3 unpaired nucleotides (hence, $i \cdot j$ cannot form if $j - i < 4$). Note that the R^s recursion serves as an efficiency wrapper of the R^b recursion (here, representing the 3′-most terminal base pair in an exterior loop context) to reduce the time complexity of the $R^{\varnothing}$ recursion from $O(N^4)$ to $O(N^3)$. This time complexity reduction is achieved by defining the 3′-most base pair using R^b within the R^s efficiency wrapper rather than directly using the R^b recursion within the $R^{\varnothing}$ recursion, so as to avoid introducing a fourth independent index into the $R^{\varnothing}$ recursion.

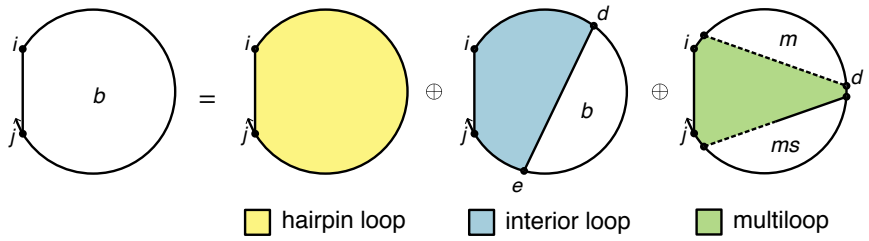
R^b_{INTRA} recursion without coaxial and dangle stacking. The $R^s_{\text{INTRA}}(i,j,\phi)$ recursion references Q^b elements that are computed using the R^b_{INTRA} recursion displayed in Figure S5. $R^b_{\text{INTRA}}(i,j,\phi)$ operates on a conditional ensemble for subsequence $[i,j]$ with i and j base paired to each other (depicted with a solid line between i and j). The function $\text{COMPLEMENTARY}(\phi_i,\phi_j)$ checks if bases ϕ_i and ϕ_j are complementary (Watson–Crick or wobble pair) without regard to whether i and j are sufficiently separated along the strand to be able to pair sterically. The recursion distinguishes three cases that are combined using $\oplus$ in the recursion equation:

- *Hairpin loop:* the hairpin loop closed by the single base pair $i \cdot j$ (depicted by a straight solid line). The recursion incorporates the recursion energy $\Delta G^{\text{hairpin}}_{i,j}(\phi)$. The index

limits in the recursion equation reflect the fact that steric constraints prevent a hairpin loop with fewer than 3 unpaired nucleotides (hence, $i \cdot j$ cannot form if $j - i < 4$).

- *Interior loop:* the interior loop closed by the two terminal base pairs $i \cdot j$ and $d \cdot e$ (depicted by straight solid lines). We defer discussion of the calculation of the interior loop contributions using the subroutine InteriorIntra until Section S2.4, where we describe both $O(N^4)$ and $O(N^3)$ recursions. The index limits in the recursion equation reflect the fact that steric effects prevent an interior loop with $j - i < 6$ due to the steric requirement that there be at least 3 intervening bases between d and e.

- *Multiloop:* the multiloop closed by three or more terminal base pairs: 1) the terminal base pair $i \cdot j$ depicted by a straight solid line, 2) a 3′-most terminal base pair starting at d and ending in the interval $[d + 1, j - 1]$ (depicted by a straight half-solid/half dashed line between d and $j - 1$); the contribution of subsequence $[d, j - 1]$ is



$$R^{b}_{\text{Intra}}(i,j,\phi) \equiv \begin{cases} C_1, & \textsc{Complementary}(\phi_i, \phi_j) \\ \mathbb{0}, & \text{otherwise} \end{cases}$$

$$\text{where } C_1 \equiv \begin{cases} W(\Delta G^{\text{hairpin}}_{i,j}(\phi)), & j - i \geq 4 \\ \mathbb{0}, & \text{otherwise} \end{cases}$$

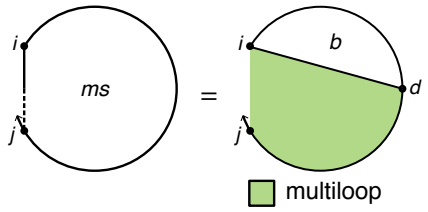
$$\oplus \begin{cases} \textsc{InteriorIntra}(i,j,\phi), & j - 1 \geq 6 \\ \mathbb{0}, & \text{otherwise} \end{cases}$$

$$\oplus \begin{cases} \textsc{dot}\left(Q^{m}_{i+1,\overline{d}}, Q^{ms}_{\overline{d}+1,j-1}\right) \otimes W(\Delta G^{\text{multi}}_{\text{init}} + \Delta G^{\text{multi}}_{\text{bp}} + \Delta G^{\text{terminalbp}}_{j,i}(\phi)), & j - i \geq 11 \\ \mathbb{0}, & \text{otherwise} \end{cases}$$

with $\overline{d} \equiv [i + 5 : j - 6]$.

Figure S5: R^{b}_{Intra} recursion without coaxial and dangle stacking. Top: recursion diagram. Bottom: recursion equation.

incorporated by element $Q^{ms}_{d,j-1}$, 3) one or more additional terminal base pairs in the interval $[i+1, d-1]$ (the straight dashed line denotes that $i+1$ and $d-1$ may or may not be paired); the contribution of subsequence $[i+1, d-1]$ is incorporated by element $Q^{m}_{i+1,d-1}$. Shading corresponds to three recursion energies: 1) the penalty for formation of a multiloop $\Delta G^{\text{multi}}_{\text{init}}$, 2) the sequence-independent penalty for a terminal base pair in a multiloop $\Delta G^{\text{multi}}_{\text{bp}}$ (corresponding to the sole base pair $i \cdot j$ that is fully defined in this recursion), 3) the sequence-dependent penalty for a terminal base pair in a multiloop context, $\Delta G^{\text{terminalbp}}_{j,i}(\phi)$ (note that the indices are ordered j then i to reflect 5′ to 3′ from the perspective of the multiloop). The index limits in the recursion equation reflect the fact that steric effects prevent a multiloop with $j - i < 11$ due to the steric requirement that there be at least 3 intervening bases between $i+1$ and d and at least 3 intervening bases between $d+1$ and $j-1$.



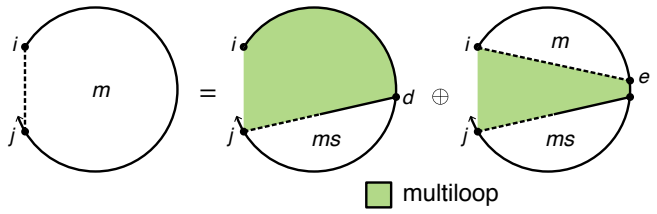
$$R^{ms}_{\text{INTRA}}(i,j,\phi) \equiv \begin{cases} \text{DOT}\left(Q^{b}_{i,\overline{d}}, W(\Delta G^{\text{multi}}_{\text{bp}} + \overline{n}_{\text{nt}}\Delta G^{\text{multi}}_{\text{nt}} + \Delta G^{\text{terminalbp}}_{i,\overline{d}}(\phi))\right), & j-i \geq 4 \\ \mathbb{0}, & \text{otherwise} \end{cases}$$

where $\overline{d} \equiv [i+4 : j]$, $\quad \overline{n}_{\text{nt}} \equiv [0 : j-i-4]^{r}$.

Figure S6: R^{ms}_{INTRA} recursion without coaxial and dangle stacking. Top: recursion diagram. Bottom: recursion equation.

R^{ms}_{INTRA} recursion without coaxial and dangle stacking. The $R^{b}_{\text{INTRA}}(i,j,\phi)$ recursion references Q^{ms} elements that are computed using the $R^{ms}_{\text{INTRA}}(i,j,\phi)$ recursion shown in Figure S6. $R^{ms}_{\text{INTRA}}(i,j,\phi)$ operates on a conditional ensemble for subsequence $[i,j]$ in a multiloop context containing one terminal base pair starting at i and ending in the interval $[i+1, j]$ (depicted as a half-solid/half-dashed line between i and j). The contribution for the subsequence $[i,d]$ enclosed by base pair $i \cdot d$ is incorporated using a $Q^{b}_{i,d}$ element. Shading corresponds to three recursion energies: 1) the sequence-independent penalty for a terminal base pair in a multiloop $\Delta G^{\text{multi}}_{\text{bp}}$ (base pair $i \cdot d$), 2) the penalty per unpaired nucleotide in

a multiloop, $\Delta G_{\text{nt}}^{\text{multi}}$ (nucleotides $d+1, \ldots, j$ for a total of $j-d$ unpaired nucleotides; as a result, this term is zeroed out in the edge case where $d = j$), 3) the sequence-dependent penalty for a terminal base pair in a multiloop context, $\Delta G_{i,d}^{\text{terminalbp}}(\phi)$ (dependent on the sequence of base pair $i \cdot d$). Note that in the dot product the range multiplying $\Delta G_{\text{nt}}^{\text{multi}}$ runs in reverse order because the number of unpaired nucleotides, $j-d$, decreases in size as d increases in size. The index limits in the recursion equation reflect the steric requirement that there be at least 3 intervening bases between i and d. Note that R^{ms} serves as an efficiency wrapper for R^b in the multiloop context in a completely analogous manner to R^s serving as an efficiency wrapper for R^b in an exterior loop context, with R^b representing the 3′-most terminal base pair in either context.



$$R^m_{\text{INTRA}}(i,j,\phi) \equiv \begin{cases} \text{DOT}\left(Q^{ms}_{\overline{d},j}, W(\overline{n}_{\text{nt}}\Delta G^{\text{multi}}_{\text{nt}})\right), & j-i \geq 4 \\ \mathbb{0}, & \text{otherwise} \end{cases}$$
$$\oplus \begin{cases} \text{DOT}\left(Q^{m}_{i,\overline{e}}, Q^{ms}_{\overline{e}+1,j}\right), & j-i \geq 9 \\ \mathbb{0}, & \text{otherwise} \end{cases}$$
$$\text{where } \overline{d} \equiv [i : j-4], \quad \overline{n}_{\text{nt}} \equiv [0 : j-i-4], \quad \overline{e} \equiv [i+4 : j-5].$$

Figure S7: R^m_{INTRA} recursion without coaxial and dangle stacking. Top: recursion diagram. Bottom: recursion equation.

R^m_{INTRA} recursion without coaxial and dangle stacking. The $R^b_{\text{INTRA}}(i,j,\phi)$ recursion references Q^m elements that are computed using the R^m_{INTRA} recursion shown in Figure S7. $R^m_{\text{INTRA}}(i,j,\phi)$ operates on a conditional ensemble for subsequence $[i,j]$ in a multiloop context where i and j may or may not be paired (depicted with a dashed line between i and j in the recursion diagram) and where there is at least one terminal base pair. This recursion distinguishes two cases that are combined using $\oplus$ in the recursion equation:

- *One terminal base pair:* the case where there is exactly one terminal base pair in subsequence $[i,j]$ in a multiloop context. This terminal base pair starts at d and ends

in the interval $[d+1, j]$ (depicted by a straight half-solid/half dashed line between d and j); the contribution of subsequence $[d, j]$ is incorporated by element $Q^{ms}_{d,j}$. Shading corresponds to the recursion energy, $\Delta G^{\text{multi}}_{\text{nt}}$, representing the penalty per unpaired nucleotide in a multiloop (nucleotides $i, \ldots, d-1$ for a total of $d-i$ unpaired nucleotides; as a result, this term is zeroed out in the edge case where $d = i$). The index limits in the recursion equation reflect the steric requirement that there be at least 3 intervening bases between d and j.

- *More than one terminal base pair:* the case where there are two or more terminal base pairs in subsequence $[i, j]$ in a multiloop context. The 3′-most terminal base pair starts at $e+1$ and ends in the interval $[e+2, j]$ (depicted by a straight half-solid/half dashed line between $e+1$ and j); the contribution of subsequence $[e+1, j]$ is incorporated by element $Q^{ms}_{e+1,j}$. There are one or more additional terminal base pairs in the interval $[i, e]$ (the straight dashed line denotes that i and e may or may not be paired); the contribution of subsequence $[i, e]$ is incorporated by element $Q^{m}_{i,e}$. The shading does not represent any recursion energies as all multiloop contributions are handled by other recursions: 1) there are no terminal base pairs in a multiloop context explicitly defined in this case, 2) there are no unpaired bases in a multiloop context explicitly defined in this case. The index limits in the recursion equation reflect the steric requirement that there be at least 3 intervening bases between i and e and at least 3 intervening bases between $e+1$ and j.

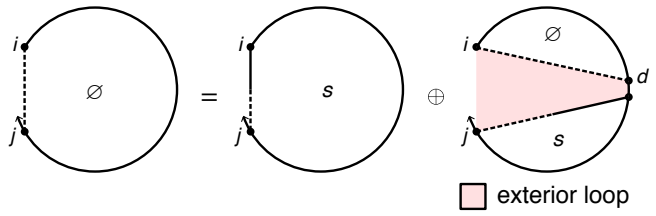
S2.3.2 Interstrand dynamic programming recursions without coaxial and dangle stacking

Here, we consider recursions for calculating the entries in a rectangular interstrand block without coaxial and dangle stacking. By definition, interstrand blocks involve 2 or more strands, and hence one or more nicks between strands. For a given interstrand block, η stores an array of nick indices between strands within the block, with each nick denoted by the index of the nucleotide following the nick. If $m \equiv \textsc{First}(\eta)$ and $n \equiv \textsc{Last}(\eta)$, then for subsequence $[i, j]$ corresponding to element i, j in the interstrand block, we have by definition $i < m$ (nucleotide i is on the first strand in the block) and $j \geq n$ (nucleotide j is on the last strand in the block).

$R^{\varnothing}_{\textsc{Inter}}$ recursion without coaxial and dangle stacking. We begin with $R^{\varnothing}_{\textsc{Inter}}(i, j, \phi)$ shown in Figure S8. $R^{\varnothing}_{\textsc{Inter}}(i, j, \phi)$ operates on the unconstrained ensemble for subsequence

$[i, j]$ with i and j on different strands in an exterior loop context where i and j may or may not be paired (depicted with a dashed line between i and j in the recursion diagram). Unlike $R^{\varnothing}_{\text{INTRA}}(i, j, \phi)$, there is no empty case because this would correspond to a disconnected structure (which is not in the multistranded ensemble) due to the presence of one or more nicks between i and j. Hence, the only case is *at least one terminal base pair:* the non-empty case in an exterior loop context where there is at least one terminal base pair (i.e., a base pair terminating a duplex) in subsequence $[i, j]$. The 3′-most terminal base pair begins at $d + 1$ and ends in the interval $[d + 2, j]$ (depicted using a half-solid/half-dashed line in the recursion diagram). The contributions for subsequence $[d+1, j]$ are incorporated using a $Q^{s}_{d+1,j}$ element. Contributions for the remaining subsequence $[i, d]$ are incorporated by a $Q^{\varnothing}_{i,d}$ element. The shading denotes the recursion energy 0 corresponding to the zero reference state in an exterior loop context. Note that the recursion energy $\Delta G^{\text{terminalbp}}(\phi)$ representing one component of the $\Delta G^{\text{exterior}}_{i,j}(\phi)$ free energy is not incorporated here because the full identity of the terminal base pair (i.e., a base pair terminating a duplex) beginning at $d + 1$ is not known within the $R^{\varnothing}_{\text{INTER}}(i, j, \phi)$ recursion (only within the $R^{s}(d + 1, j, \phi)$ recursion). The edge case where the index $d + 1 = i$ is displayed explicitly to indicate that no $Q^{\varnothing}$ element is accessed in this case.

Because there are nicks involved in calculating the elements of interstrand blocks, care must be taken to ensure that no disconnected secondary structures are incorporated in the complex ensemble. For a given interstrand block with nick indices η, the function VALID returns the set of valid vectorization ranges $\{\overline{d}_1, \overline{d}_2, \dots\}$, such that for each valid vectorization range, d and $d + 1$ are on the same strand (i.e., such that d and $d + 1$ do not take on values that would



$$R^{\varnothing}_{\text{INTER}}(i, j, \phi) \equiv Q^{s}_{i,j} \oplus \bigoplus_{\overline{d} \in \text{VALID}(i, \max(j-4, n), \eta)} \text{DOT}\left(Q^{\varnothing}_{i,\overline{d}}, Q^{s}_{\overline{d}+1,j}\right) \tag{S44}$$

where $n = \text{LAST}(\eta)$

Figure S8: $R^{\varnothing}_{\text{INTER}}$ recursion without coaxial and dangle stacking. Top: recursion diagram. Bottom: recursion equation.

place a nick between them). As is evident from the recursion diagram of Figure S8, if d and $d+1$ were to take on values that placed a nick between them, a disconnected structure would result. There is at most one valid vectorization range per strand, and there may be none for a strand or subsequence that is too short. For each valid vectorization range $\overline{d}$, the resulting dot product

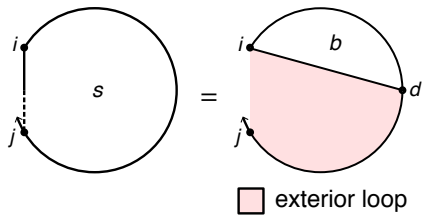
$$\textsc{dot}\left(Q^{\varnothing}_{i,\overline{d}}, Q^{s}_{\overline{d}+1,j}\right) \tag{S45}$$

is between the range $\overline{d}$ of row i (depicted as brown elements in Figure S1c) and the range $\overline{d}+1$ of column j (gray elements), yielding element i, j (purple element). Note that Figure S1c depicts two valid vectorization ranges (leading to two dot products that are summed to calculate the purple element); the gap of one element between the two vectorization ranges corresponds to exclusion of the value $d = 3$ which would have placed a nick between nucleotides d and $d+1$ (note that $\eta = 4$ for this interstrand block).

Note that for calculating element i, j in Figure S8, the subsequence submitted to Valid ranges from i to $\max(j-4, n)$, where $n \equiv \textsc{Last}(\eta)$. This yields two cases:

- If $\max(j-4, n) = j-4$: there is no nick between nucleotide $j-4$ and j (since $n \equiv \textsc{Last}(\eta) < j-4$), so there must be at least 3 intervening bases between $d+1$ and j because steric effects prevent a hairpin loop with fewer than 3 unpaired nucleotides. In this case, each incorporated element $Q^{s}_{d+1,j}$ results from an $R^{s}_{\text{Intra}}(d+1, j, \phi)$ recursion for an intrastrand block.
- If $\max(j-4, n) = n$: there is a nick between nucleotide $j-4$ and j (since $n \geq j-4$), so $d+1$ can be as large as $n-1$ and still pair to any nucleotide in subsequence $[n, j]$. In this case, each incorporated element $Q^{s}_{d+1,j}$ results from an $R^{s}_{\text{Inter}}(d+1, j, \phi)$ recursion for an interstrand block.

R^{s}_{Inter} recursion without coaxial and dangle stacking. The $R^{\varnothing}_{\text{Intra}}(i, j, \phi)$ recursion references $Q^{s}_{d+1,j}$ elements that are computed using either the R^{s}_{Intra} recursion of Figure S4 (if $d+1$ and j are on the same strand) or the R^{s}_{Inter} recursion of Figure S9 (if $d+1$ and j are on different strands). Recursion $R^{s}_{\text{Inter}}(i, j, \phi)$ operates on a conditional ensemble for subsequence $[i, j]$ with i and j on different strands in an exterior loop context containing one terminal base pair starting at i and ending in the interval $[i+1, j]$ (depicted as a half-solid/half-dashed line between i and j). The contribution for the subsequence $[i, d]$ enclosed by base pair $i \cdot d$ is incorporated using a $Q^{b}_{i,d}$ element. Shading corresponds to



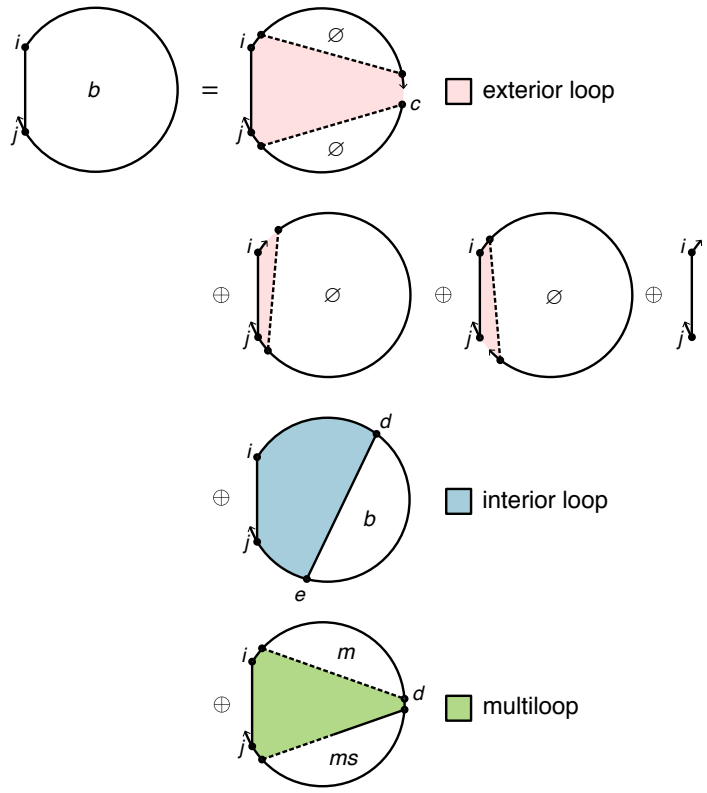
$$R^{s}_{\text{INTER}}(i,j,\phi) \equiv \text{DOT}\left(Q^{b}_{i,\overline{d}}, W(\Delta G^{\text{terminalbp}}_{i,\overline{d}}(\phi))\right), \quad \text{where } \overline{d} \equiv [\text{LAST}(\eta) : j] \tag{S46}$$

Figure S9: R^{s}_{INTER} recursion without coaxial and dangle stacking. Top: recursion diagram. Bottom: recursion equation.

the recursion energy, $\Delta G^{\text{terminalbp}}_{i,d}(\phi)$, representing the sequence-dependent penalty for a terminal base pair in an exterior loop context. The index d must always be on the last strand (i.e., $d \geq \text{LAST}(\eta)$) to ensure there are no strand breaks in the subsequence $[d, j]$, which would correspond to a disconnected structure. Note that the R^{s} recursion serves as an efficiency wrapper of the R^{b} recursion (here, representing the 3′-most terminal base pair in an exterior loop context) to reduce the time complexity of the $R^{\varnothing}$ recursion from $O(N^4)$ to $O(N^3)$. This time complexity reduction is achieved by defining the 3′-most terminal base pair using R^{b} within the R^{s} efficiency wrapper rather than directly using the R^{b} recursion within the $R^{\varnothing}$ recursion, so as to avoid introducing a fourth independent index into the $R^{\varnothing}$ recursion.

R^{b}_{INTER} recursion without coaxial and dangle stacking. The $R^{s}_{\text{INTRA}}(i,j,\phi)$ recursion references $Q^{b}_{i,d}$ elements that are computed using either the R^{b}_{INTRA} recursion of Figure S10 (if i and d are on the same strand) or the R^{b}_{INTER} recursion of Figure S10 (if i and d are on different strands). $R^{b}_{\text{INTER}}(i,j,\phi)$ operates on a conditional ensemble for subsequence $[i,j]$ with i and j on different strands and base paired to each other (depicted with a solid line between i and j). The function $\text{COMPLEMENTARY}(\phi_i, \phi_j)$ checks if bases ϕ_i and ϕ_j are complementary (Watson–Crick or wobble pair) without regard to whether i and j are sufficiently separated along the strand to be able to pair sterically. The recursion distinguishes three cases that are combined using $\oplus$ in the recursion equation:

- *Exterior loop:* the exterior loop closed by one or more terminal base pairs including terminal base pair $i \cdot j$.



$$R^{b}_{\text{INTER}}(i,j,\phi) \equiv \begin{cases} C_1, & \textsc{Complementary}(\phi_i,\phi_j) \\ \mathbb{0}, & \text{otherwise} \end{cases}$$

$$C_1 \equiv \begin{cases} \bigoplus_{c\in\eta} Q^{\varnothing}_{i+1,c-1} \otimes Q^{\varnothing}_{c,j-1} \otimes W(\Delta G^{\text{terminalbp}}_{j,i}(\phi)), & i+1 \neq m \text{ and } j \neq n \\ Q^{\varnothing}_{m,j-1} \otimes W(\Delta G^{\text{terminalbp}}_{j,i}(\phi)), & i+1 = m \text{ and } j \neq n \\ Q^{\varnothing}_{i+1,n-1} \otimes W(\Delta G^{\text{terminalbp}}_{j,i}(\phi)), & i+1 \neq m \text{ and } j = n \\ W(\Delta G^{\text{terminalbp}}_{j,i}(\phi)), & i+1 = j = m = n \end{cases}$$

$$\oplus\ \textsc{InteriorInter}(i,j,\phi)$$

$$\oplus \bigoplus_{\overline{d}\in\textsc{Valid}(i+1,j-1,\eta)} \textsc{dot}\left(Q^{m}_{i+1,\overline{d}}, Q^{ms}_{\overline{d}+1,j-1}\right) \otimes W(\Delta G^{\text{multi}}_{\text{init}} + \Delta G^{\text{multi}}_{\text{bp}} + \Delta G^{\text{terminalbp}}_{j,i}(\phi))$$

where $m = \textsc{First}(\eta)$

$n = \textsc{Last}(\eta)$

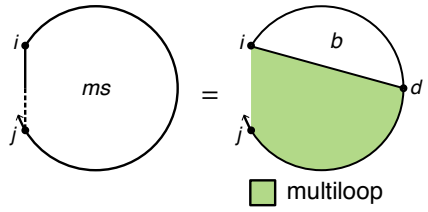
Figure S10: R^{b}_{INTER} recursion without coaxial and dangle stacking. Top: recursion diagram. Bottom: recursion equation.

  - *Base case:* The base case corresponds to the recursion diagram in the first row of Figure S10 with a nick at c. For each nick $c \in \eta$, the contribution of subsequence $[i+1, c-1]$ is incorporated by element $Q^{\varnothing}_{i+1,c-1}$ and the contribution of subsequence $[c, j-1]$ is incorporated by element $Q^{\varnothing}_{c,j-1}$. Shading corresponds to the recursion energy $\Delta G^{\text{terminalbp}}_{j,i}(\phi)$ representing the sequence-dependent penalty for a terminal base pair in an exterior loop context, (note that the indices are ordered j then i to reflect 5′ to 3′ from the perspective of the exterior loop).
  - *Edge cases:* In the base case, there is a $Q^{\varnothing}$ element on either side of the nick. In the edge cases treated by the three diagrams in the second row of Figure S10, one or both of these subsequences is absent because the nick is adjacent to i (diagram 1), adjacent to j (diagram 2), or adjacent to both i and j (diagram 3).

- *Interior loop:* the interior loop closed by the two terminal base pairs $i \cdot j$ and $d \cdot e$ (depicted by straight solid lines). We defer discussion of the calculation of the interior loop contributions using InteriorInter until Section S2.4, where we describe both $O(N^4)$ and $O(N^3)$ recursions.

- *Multiloop:* the multiloop closed by three or more terminal base pairs: 1) the terminal base pair $i \cdot j$ depicted by a straight solid line, 2) a 3′-most terminal base pair starting at $d+1$ and ending in interval $[d+2, j-1]$ (depicted by a straight half-solid/half dashed line between $d+1$ and $j-1$); the contribution of subsequence $[d+1, j-1]$ is incorporated by element $Q^{ms}_{d+1,j-1}$, 3) one or more additional terminal base pairs in the interval $[i+1, d]$ (the straight dashed line denotes that $i+1$ and d may or may not be paired); the contribution of subsequence $[i+1, d]$ is incorporated by element $Q^{m}_{i+1,d}$. Shading corresponds to three recursion energies: 1) the penalty for formation of a multiloop $\Delta G^{\text{multi}}_{\text{init}}$, 2) the sequence-independent penalty for a terminal base pair in a multiloop $\Delta G^{\text{multi}}_{\text{bp}}$ (corresponding to the sole base pair $i \cdot j$ that is fully defined in this recursion), 3) the sequence-dependent penalty for a terminal base pair in a multiloop context, $\Delta G^{\text{terminalbp}}_{j,i}(\phi)$ (note that the indices are ordered j then i to reflect 5′ to 3′ from the perspective of the multiloop). To exclude exterior loop states that are not treated by this multiloop recursion, the function Valid returns the set of valid vectorization ranges for which nucleotides d and $d+1$ are on the same strand (i.e., such that d and $d+1$ do not take on values that would place a nick between them).

Note that unlike the R^b_{Intra} recursion of Figure S5, for R^b_{Inter} there is no hairpin loop case as i and j are on different strands.



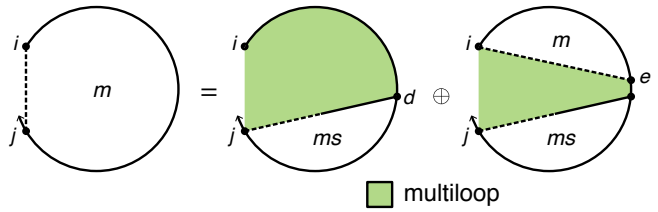
$$R^{ms}_{\text{INTER}}(i,j,\phi) \equiv \text{DOT}\left(Q^{b}_{i,\overline{d}}, W(\Delta G^{\text{multi}}_{\text{bp}} + \overline{n}_{\text{nt}}\Delta G^{\text{multi}}_{\text{nt}} + \Delta G^{\text{terminalbp}}_{i,\overline{d}}(\phi))\right)$$

$$\text{where } \overline{d} \equiv [\text{LAST}(\eta) : j], \quad \overline{n}_{\text{nt}} \equiv [0 : j - \text{LAST}(\eta)]^r$$

Figure S11: R^{ms}_{INTER} recursion without coaxial and dangle stacking. Top: recursion diagram. Bottom: recursion equation.

R^{ms}_{INTER} recursion without coaxial and dangle stacking. The $R^{b}_{\text{INTER}}(i,j,\phi)$ recursion references $Q^{ms}_{d+1,j-1}$ elements that are computed using either the R^{ms}_{INTRA} recursion shown of Figure S6 (if $d+1$ and $j-1$ are on the same strand) or the R^{ms}_{INTER} recursion of Figure S11 (if $d+1$ and $j-1$ are on different strands). $R^{ms}_{\text{INTER}}(i,j,\phi)$ operates on a conditional ensemble for subsequence $[i,j]$ in a multiloop context containing one terminal base pair starting at i and ending in the interval $[i+1,j]$ (depicted as a half-solid/half-dashed line between i and j). The contribution for the subsequence $[i,d]$ enclosed by base pair $i \cdot d$ is incorporated using a $Q^{b}_{i,d}$ element. Shading corresponds to three recursion energies: 1) the sequence-independent penalty for a terminal base pair in a multiloop, $\Delta G^{\text{multi}}_{\text{bp}}$ (base pair $i \cdot d$), 2) the penalty per unpaired nucleotide in a multiloop, $\Delta G^{\text{multi}}_{\text{nt}}$ (nucleotides $d+1,\ldots,j$ for a total of $j-d$ unpaired nucleotides; as a result, this term is zeroed out in the edge case where $d = j$), 3) the sequence-dependent penalty for a terminal base pair in a multiloop context, $\Delta G^{\text{terminalbp}}_{i,d}(\phi)$ (dependent on the sequence of base pair $i \cdot d$). Note that in the dot product the range multiplying $\Delta G^{\text{multi}}_{\text{nt}}$ runs in reverse order because the number of unpaired nucleotides, $j-d$, decreases in size as d increases in size. Nucleotide d must always be on the last strand to ensure that there are no nicks in the subsequence $[d,j]$, which would lead to either a disconnected structure (which is not permitted in the complex ensemble) or an exterior loop state (which is not handled by this multiloop recursion). Note that R^{ms} serves as an efficiency wrapper for R^{b} in the multiloop context in a completely analogous manner to R^{s} serving as an efficiency wrapper for R^{b} in an exterior loop context, with R^{b} representing the 3′-most terminal base pair in either context.

R^{m}_{INTER} recursion without coaxial and dangle stacking. The $R^{b}_{\text{INTER}}(i,j,\phi)$ recursion references $Q^{m}_{i+1,d}$ elements that are computed using either the R^{m}_{INTRA} recursion of Figure S7



$$R^{m}_{\text{INTER}}(i,j,\phi) \equiv \text{DOT}\left(Q^{ms}_{\overline{d},j}, W(\overline{n}_{\text{nt}}\Delta G^{\text{multi}}_{\text{nt}})\right) \oplus \bigoplus_{\overline{e}\in \text{VALID}(i,j,\eta)} \text{DOT}\left(Q^{m}_{i,\overline{e}}, Q^{ms}_{\overline{e}+1,j}\right)$$

$$\text{where } \overline{d} \equiv [i : \text{FIRST}(\eta) - 1], \quad \overline{n}_{\text{nt}} \equiv [0 : \text{FIRST}(\eta) - i - 1]$$

Figure S12: R^{m}_{INTER} recursion without coaxial and dangle stacking. Top: recursion diagram. Bottom: recursion equation.

(if $i+1$ and d are on the same strand), or the R^{m}_{INTER} recursion of Figure S12 (if $i+1$ and d are on different strands). $R^{m}_{\text{INTER}}(i,j,\phi)$ operates on a conditional ensemble for subsequence $[i,j]$ in a multiloop context where i and j may or may not be paired (depicted with a dashed line between i and j in the recursion diagram) and where there is at least one terminal base pair. This recursion distinguishes two cases that are combined using $\oplus$ in the recursion equation:

- *One terminal base pair:* the case where there is exactly one terminal base pair in subsequence $[i,j]$ in a multiloop context. This terminal base pair starts at d and ends in the interval $[d+1,j]$ (depicted by a straight half-solid/half dashed line between d and j); the contribution of subsequence $[d,j]$ is incorporated by element $Q^{ms}_{d,j}$. Shading corresponds to the recursion energy, $\Delta G^{\text{multi}}_{\text{nt}}$, representing the penalty per unpaired nucleotide in a multiloop (nucleotides $i,\ldots,d-1$ for a total of $d-i$ unpaired nucleotides; as a result, this term is zeroed out in the edge case where $d=i$). Nucleotide d must always be on the first strand to ensure that there are no nicks in the subsequence $[i,d]$, which would lead to either a disconnected structure (which is not permitted in the complex ensemble) or an exterior loop state (which is not handled by this multiloop recursion).

- *More than one terminal base pair:* the case where there are two or more terminal base pairs in subsequence $[i,j]$ in a multiloop context. The 3′-most terminal base pair starts at $e+1$ and ends in the interval $[e+2,j]$ (depicted by a straight half-solid/half dashed line between $e+1$ and j); the contribution of subsequence $[e+1,j]$ is incorporated by element $Q^{ms}_{e+1,j}$. There are one or more additional terminal base

pairs in the interval $[i, e]$ (the straight dashed line denotes that i and e may or may not be paired); the contribution of subsequence $[i, e]$ is incorporated by element $Q^m_{i,e}$. The shading does not represent any recursion energies as all multiloop contributions are handled by other recursions: 1) there are no terminal base pairs in a multiloop context explicitly defined in this case, 2) there are no unpaired bases in a multiloop context explicitly defined in this case. To exclude exterior loop states that are not treated by this multiloop recursion, the function VALID returns the set of valid vectorization ranges for which nucleotides e and $e + 1$ are on the same strand (i.e., such that e and $e + 1$ do not take on values that would place a nick between them).

S2.4 Recursions for interior loop contributions

Interior loop contributions to the recursions $R^b_{\text{INTRA}}(i, j, \phi)$ and $R^b_{\text{INTER}}(i, j, \phi)$ run naively in $O(N^4)$ time, as is evident from the four indices i, d, e, j in the interior loop recursion diagrams in Figures S5 and S10.

$O(N^4)$ intrastrand interior loop recursion. The intrastrand $O(N^4)$ interior loop contribution:

$$O(N^4)\ \text{INTERIORINTRA}(i, j, \phi) \equiv \begin{cases} \bigoplus_{d=i+1}^{j-5} \quad \bigoplus_{e=d+4}^{j-1} \{Q^b_{d,e} \otimes W(\Delta G^{\text{interior}}_{i,d,e,j}(\phi))\}, & j - i \geq 6 \\ \mathbb{0}, & \text{otherwise} \end{cases} \tag{S47}$$

considers interior loops through a nested iteration, first over d in a $5'$ to $3'$ direction and for each d over e in a $5'$ to $3'$ direction. The index limits in the recursion equation reflect the fact that steric effects prevent an interior loop with $j - i < 6$ due to the steric requirement that there be at least 3 intervening bases between d and e. The function $\Delta G^{\text{interior}}_{i,d,e,j}(\phi)$ accounts for the free energy of the loop with bounding base pairs $i \cdot j$ and $d \cdot e$, substituting in the correct functional form for any of the various interior loop types (stacked pair, bulge, etc; see Section S1.7.2).

$O(N^4)$ interstrand interior loop recursion. The interstrand $O(N^4)$ interior loop contribution:

$$O(N^4)\ \textsc{InteriorInter}(i,j,\phi) \equiv \begin{cases} \bigoplus_{d=i+1}^{m-1} \bigoplus_{e=n}^{j-1} \\ \qquad \{Q^b_{d,e} \otimes W(\Delta G^{\text{interior}}_{i,d,e,j}(\phi))\}, & i < m-1 \text{ and } n < j \\ \mathbb{0}, & \text{otherwise} \end{cases}$$
$$\text{where } m = \textsc{First}(\eta)$$
$$n = \textsc{Last}(\eta) \tag{S48}$$

proceeds in the same general manner, considering interior loops in order of ascending d then e indices. However, d is restricted to be on the first strand ($d < m$) and e is restricted to be on the last strand ($e \geq n$), as reflected in the upper summation limit for d and the lower summation limit for e. These two requirements ensure that there are no nicks between i and d and between e and j, preventing exterior loop states (that are not treated in this interior loop recursion) and disconnected states (that are not part of the complex ensemble).

$O(N^3)$ intrastrand interior loop recursion. To reduce the complexity of computing interior loop contributions from $O(N^4)$ to $O(N^3)$, we must exploit the functional form of the free energy model for large interior loops (Section S1.7.2)[11]. In References [4] and [5], this optimization was referred to as the "fastiloops" or "fast interior loops" function, and we take a similar approach here. The following optimizations assume the use of a forward operation order (not a backtracking operation order). Interior loops, defined by two bounding base pairs $i \cdot j$ and $d \cdot e$, can be classified by the distances $L_1 = d - i - 1$ and $L_2 = j - e - 1$; L_1 and L_2 are the numbers of unpaired nucleotides on each side of the interior loop. In cases where $L_1 < 4$ or $L_2 < 4$, the energy functions generally depend on terms that are nonlinear with respect to L_1 and L_2. Examples include the special-case energy functions for stacked pairs and bulge loops, as well as length-dependent asymmetry and size penalties for other interior loops. We term these interior loops *inextensible* because the free energy for a larger loop cannot in general be calculated using the value from a smaller loop. For a given subsequence $[i, j]$, there are only $O(N)$ inextensible interior loops (because of the constant upper bound on L_1 or L_2) so they do not contribute to the $O(N^4)$ complexity.

The remaining interior loops in which $L_1 \geq 4$ and $L_2 \geq 4$ are referred to as *extensible* interior loops because the free energy of a larger loop can be calculated by extending the calculation from a smaller interior loop. For a given subsequence $[i, j]$, there are $O(N^2)$ extensible interior loops so these are the cases we must deal with efficiently to reduce the time complexity from $O(N^4)$ to $O(N^4)$. For extensible interior loops, (S28) gives:

$$\Delta G^{\text{interior}}_{i,d,e,j}(\phi) = \Delta G^{\text{interiorsize}}_{L_1+L_2} + \Delta G^{\text{interiorasymm}}_{|L_1-L_2|} + \Delta G^{\text{interiormm}}_{j-1,j,i,i+1}(\phi) + \Delta G^{\text{interiormm}}_{d-1,d,e,e+1}(\phi). \quad \text{(S49)}$$

Here, the quantity $\Delta G^{\text{interiorsize}}_{L_1+L_2}$ is a sequence-independent free energy contribution due to the size of the interior loop, $s \equiv L_1 + L_2$ (the sum of the two side lengths). The quantity $\Delta G^{\text{interiorasymm}}_{|L_1-L_2|}$ is a sequence-independent free energy contribution due to the *asymmetry* of the loop, $|L_1 - L_2|$ (the difference of the two side lengths). Finally, the two terms $\Delta G^{\text{interiormm}}_{j-1,j,i,i+1}(\phi)$ and $\Delta G^{\text{interiormm}}_{d-1,d,e,e+1}(\phi)$ are sequence-dependent free energy contributions due to mismatch stacking on the base pairs $i \cdot j$ and $d \cdot e$, respectively.

Two key insights from (S49) allow us to use this functional form to reduce complexity.[11] First, for every base pair $i \cdot j$, the mismatch term for that base pair is independent of the other quantities and can be factored out. Second, for a given base pair $d \cdot e$, an extensible loop bounded by $i \cdot j$ can be converted to an extensible loop bounded by $i - 1 \cdot j + 1$ by updating $\Delta G^{\text{interiorsize}}_{s}$ to $\Delta G^{\text{interiorsize}}_{s+2}$ and replacing $\Delta G^{\text{interiormm}}_{j-1,j,i,i+1}(\phi)$ with $\Delta G^{\text{interiormm}}_{j,j+1,i-1,i}(\phi)$. Thus, we can cache the information specific to the base pair $d \cdot e$ for each given asymmetry the first time it is encountered in an extensible interior loop and then modify only the size information each time it is encountered.

Equation S50 combines the above ideas into a subroutine for computing the interior loop contributions to $R^b_{\text{INTRA}}(i, j, \phi)$.

$$
\begin{aligned}
O(N^3)\ \textsc{InteriorIntra}(i,j,\phi) \equiv & \begin{cases} \bigoplus_{d=i+1}^{\min(i+4,j-5)} \bigoplus_{e=\max(d+4,j-4)}^{j-1} & \\ \qquad Q^b_{d,e} \otimes W(\Delta G^{\text{interior}}_{i,d,e,j}(\phi)), & i+6 \le j \\ \mathbb{0}, & \text{otherwise} \end{cases} \\
& \oplus \begin{cases} \bigoplus_{d=i+1}^{\min(i+4,j-9)} \bigoplus_{e=d+4}^{j-5} & \\ \qquad Q^b_{d,e} \otimes W(\Delta G^{\text{interior}}_{i,d,e,j}(\phi)), & i+10 \le j \\ \mathbb{0}, & \text{otherwise} \end{cases} \\
& \oplus \begin{cases} \bigoplus_{d=i+5}^{j-5} \bigoplus_{e=\max(d+4,j-4)}^{j-1} & \\ \qquad Q^b_{d,e} \otimes W(\Delta G^{\text{interior}}_{i,d,e,j}(\phi)), & i+10 \le j \\ \mathbb{0}, & \text{otherwise} \end{cases} \\
& \oplus \begin{cases} \bigoplus_{s=8}^{j-i-6} Q^x_{i,j,s} \otimes W(\Delta G^{\text{interiormm}}_{j-1,j,i,i+1}(\phi)), & i+14 \le j \\ \mathbb{0}, & \text{otherwise} \end{cases}.
\end{aligned}
\tag{S50}
$$

The first three rows handle inextensible interior loops for three cases: 1) $L_1 < 4$ and $L_2 < 4$, 2) $L_1 < 4$ and $L_2 \geq 4$, 3) $L_1 \geq 4$ and $L_2 < 4$. In each case, the contribution of subsequence $[d, e]$ is incorporated using a $Q^b_{d,e}$ element and the interior loop free energy, $\Delta G^{\text{interior}}_{i,d,e,j}(\phi)$, is evaluated as for the $O(N^4)$ intrastrand recursion. The fourth row handles extensible interior loops ($L_1 \geq 4$ and $L_2 \geq 4$), by combining a previously computed $Q^x_{i,j,s}$ element for each loop size s with the terminal mismatch free energy, $\Delta G^{\text{interiormm}}_{j-1,j,i,i+1}(\phi)$, corresponding to closing base pair $i \cdot j$. For all four cases, the index limits reflect there steric requirement that there be at least 3 intervening bases between d and e.

The $R^x_{\text{INTRA}}(i,j,s,\phi)$ recursion fills in the three-dimensional tensor $Q^x_{i,j,s}$:

$$R^x_{\text{INTRA}}(i,j,s,\phi) \equiv \begin{cases} C_1 \oplus C_2 \oplus C_3, & j-i > 15 \text{ and } 10 \le s \le j-i-6 \\ C_2 \oplus C_3, & j-i > 14 \text{ and } s = 9 \\ C_2, & j-i > 13 \text{ and } s = 8 \\ C_3, & j-i = 14 \text{ and } s = 9 \\ \mathbb{0}, & \text{otherwise} \end{cases}$$

$$\begin{aligned} \text{where } C_1 &\equiv Q^x_{i+1,j-1,s-2} \otimes W(\Delta G^{\text{interiorsize}}_s - \Delta G^{\text{interiorsize}}_{s-2}) \\ C_2 &\equiv Q^b_{i+5,j+3-s} \otimes W(\Delta G^{\text{interiorsize}}_s + \Delta G^{\text{interiorasymm}}_{s-8} + \Delta G^{\text{interiormm}}_{i+4,i+5,j+3-s,j+4-s}(\phi)) \\ C_3 &\equiv Q^b_{s+i-3,j-5} \otimes W(\Delta G^{\text{interiorsize}}_s + \Delta G^{\text{interiorasymm}}_{s-8} + \Delta G^{\text{interiormm}}_{s+i-4,s+i-3,j-5,j-4}(\phi)). \end{aligned} \tag{S51}$$

The indices i and j refer to the closing base pair $i \cdot j$ while the index s refers to the size of the extensible loops collected in $Q^x_{i,j,s}$. The contributions can be divided into two classes: previously encountered loops and new loops. The previously encountered loops are incorporated by accessing the previously computed element $Q^x_{i+1,j-1,s-2}$ and replacing $\Delta G^{\text{interiorsize}}_{s-2}$ with $\Delta G^{\text{interiorsize}}_s$ (see term C1). This is the key operation that reduces the complexity of the interior loop recursion to $O(N)$ by capturing all previous loops in $O(1)$. Note that the terminal mismatch contribution of the closing base pair $i \cdot j$ is not incorporated in the Q^x element, but is combined with Q^x in (S50), so there is never a need to replace one terminal mismatch contribution for another as the loop is extended. New extensible loops that are first encountered for the indices i, j, s (elements that have exactly $L_1 = 4$ or $L_2 = 4$ or both) are handled by C_2 and C_3. Note that the subexpressions C_2 and C_3 are coincident for $L_1 = L_2 = 4$ ($s = 8$).

Note that to calculate a new value $Q^x_{i,j,s}$ for a subsequence of length $l = j-i+1$, only elements of the form $Q^x_{i+1,j-1,s-2}$ are accessed (for $O(N)$ values of s; each value of l corresponds to a diagonal of the intrastrand block). Therefore, we only need to store elements of Q^x for subsequences of length $l, l-1$, and $l-2$ (corresponding to the current diagonal and the two previous diagonals). In other words, only Q^x values corresponding to 3 diagonals need to exist in memory during the forward pass. In moving to the next diagonal $l+1$, we can simply delete all $Q^x_{i,j,s}$ values for diagonal $l-2$ as they will not be accessed again. Hence, only $O(N^2)$ space is necessary to store the needed elements of Q^x. Naively storing all of $Q^x_{i,j,s}$ would have needlessly increased the space complexity to $O(N^3)$.

$O(N^3)$ interstrand interior loop recursion. Interior loop contributions for elements in interstrand blocks are computed with $O(N^3)$ time complexity using the subroutine:

$$
\begin{aligned}
O(N^3)\ \textsc{InteriorInter}(i,j,\phi) \equiv & \begin{cases} \bigoplus_{d=i+1}^{\min(i+4,m-1)} \bigoplus_{e=\max(n,j-4)}^{j-1} & \\ \qquad Q^b_{d,e} \otimes W(\Delta G^{\text{interior}}_{i,d,e,j}(\phi)), & i+1<m \text{ and } n<j \\ \mathbb{0}, & \text{otherwise} \end{cases} \\
\oplus & \begin{cases} \bigoplus_{d=i+1}^{\min(i+4,m-1)} \bigoplus_{e=n}^{j-5} & \\ \qquad Q^b_{d,e} \otimes W(\Delta G^{\text{interior}}_{i,d,e,j}(\phi)), & i+1<m \text{ and } n+4<j \\ \mathbb{0}, & \text{otherwise} \end{cases} \\
\oplus & \begin{cases} \bigoplus_{d=i+5}^{m-1} \bigoplus_{e=\max(n,j-4)}^{j-1} & \\ \qquad Q^b_{d,e} \otimes W(\Delta G^{\text{interior}}_{i,d,e,j}(\phi)), & i+5<m \text{ and } n<j \\ \mathbb{0}, & \text{otherwise} \end{cases} \\
\oplus & \begin{cases} \bigoplus_{s=8}^{j-n+m-i-3} & \\ \qquad Q^x_{i,j,s} \otimes W(\Delta G^{\text{interiormm}}_{j-1,j,i,i+1}(\phi)), & i+5<m \text{ and } n+4<j \\ \mathbb{0}, & \text{otherwise} \end{cases}
\end{aligned}
$$

$$
\begin{aligned}
\text{where } m &= \textsc{First}(\eta) \\
n &= \textsc{Last}(\eta).
\end{aligned}
\tag{S52}
$$

The approach is analogous to that of equation S50. Index limits are modified to ensure that d is on the same strand as i and e is on the same strand as j, preventing exterior loop states (that are not treated in this interior loop recursion) and disconnected states (that are not part of the complex ensemble).

The recursion $R^x_{\text{INTER}}(i, j, s, \phi)$ is also closely related to equation S51:

$$R^x_{\text{INTER}}(i,j,s,\phi) \equiv \begin{cases} C_1 + C_2 + C_3, & i+6 < m \text{ and } n+5 < j \text{ and } 10 \le s \le j-i+m-n-3 \\ C_2 + C_3, & i+6 < m \text{ and } n+5 < j \text{ and } s = 9 \\ C_3, & i+6 < m \text{ and } n+5 = j \text{ and } s = 9 \\ C_2, & i+6 = m \text{ and } n+5 \le j \text{ and } s = 9 \\ C_2, & i+6 \le m \text{ and } n+5 \le j \text{ and } s = 8 \\ \mathbb{0}, & \text{otherwise} \end{cases}$$

$$\begin{aligned} \text{where } m &= \textsc{First}(\eta) \\ n &= \textsc{Last}(\eta) \\ C_1 &\equiv Q^x_{i+1,j-1,s-2} \otimes W(\Delta G^{\text{interiorsize}}_s - \Delta G^{\text{interiorsize}}_{s-2}) \\ C_2 &\equiv Q^b_{i+5,j+3-s} \otimes W(\Delta G^{\text{interiorsize}}_s + \Delta G^{\text{interiorasymm}}_{s-8} + \Delta G^{\text{interiormm}}_{i+4,i+5,j+3-s,j+4-s}(\phi)) \\ C_3 &\equiv Q^b_{s+i-3,j-5} \otimes W(\Delta G^{\text{interiorsize}}_s + \Delta G^{\text{interiorasymm}}_{s-8} + \Delta G^{\text{interiormm}}_{s+i-4,s+i-3,j-5,j-4}(\phi)) \end{aligned} \tag{S53}$$

The recursive component that extends previously encountered extensible loops is shown in C_1. Newly encountered extensible loops (elements that have exactly $L_1 = 4$ or $L_2 = 4$ or both) are handled by C_2 and C_3. Note that C_2 and C_3 are coincident for $L_1 = L_2 = 4$ ($s = 8$). The conditional checks using m and n prevent exterior loop states (that are not treated in this interior loop recursion) and disconnected states (that are not part of the complex ensemble).

The above recursions enable calculation of interior loop contributions for forward algorithms with $O(N^3)$ time complexity and $O(N^2)$ space complexity. However, this approach is incompatible with backtracking algorithms as the optimization of throwing away Q^x values that are no longer needed during the forward sweep, implies that they are also no longer available for backtracking after the forward sweep is complete. One option is to reconstruct the Q^x values during backtracking, but this incurs $O(N^3)$ time complexity and can lead to loss of precision for large complex ensembles [5]. Another option that we pursue here is to use a different iteration pattern through the $O(N^4)$ interior loop recursions during backtracking. With this option, we exploit the fact that unlike forward algorithms that evaluate recursive elements for all i and j in a forward sweep, backtracking algorithms evaluate only a subset of all possible recursive elements. Hence, as discussed in Section S4.4, the worst-case time complexity can be kept at $O(N^2)$ per structure for our backtracking algorithms.

S2.5 Approximate dangle stacking without coaxial stacking (for backwards compatibility with NUPACK 3)

Previous versions of NUPACK algorithms did not support coaxial stacking and offered two approximate treatments of dangle stacking (`some-nupack3` and `all-nupack3`)[7]. For backwards compatibility, NUPACK 4.0 supports these two options. A nucleotide in a multiloop or an exterior loop is eligible to dangle stack on an adjacent base pair that is either 5′ or 3′ of the nucleotide. The NUPACK 4.0 model appropriately Boltzmann-weights these two competing dangle stacking states. The NUPACK 3.2 model either: (1) took the MFE of these two dangle stacking states – as if only the MFE dangle stack occurs at equilibrium (`some-nupack3` option), or (2) summed the free energies of the two dangle stacking states – as if both dangle stacking states were occurring at once (`all-nupack3` option). These approximate dangle treatments are implemented in the NUPACK 4.0 code base using modified versions of $R^a_{\text{INTRA}}(i,j,\phi)$ and $R^a_{\text{INTER}}(i,j,\phi)$ for $a \in \{\varnothing, s, m, ms\}$. In these approximate dangle treatments (`some-nupack3` or `all-nupack3`), if dangles stack on an adjacent base pair from both the 5′ and 3′ sides at once, both dangle free energies are incorporated in lieu of incorporating a terminal mismatch free energy (equation (S55)).

S2.6 Recursions with coaxial and dangle stacking subensembles

Here, we describe $R^a_{\text{INTRA}}(i,j,\phi)$ recursions for calculating the elements of intrastrand blocks and $R^a_{\text{INTER}}(i,j,\phi)$ recursions for calculating the elements of interstrand blocks for the complex ensemble, $\overline{\Gamma}^{||}$, including coaxial and dangle stacking subensembles. For the previously defined exterior loop and multiloop recursions without coaxial and dangle stacking (see Section S2.3), the elementary recursion entity was a *terminal base pair* (a base pair that terminates a duplex to form a part of the exterior loop or multiloop). For example, a recursion might contain exactly one terminal base pair, a 3′-most terminal base pair, or one or more terminal base pairs. Here, for exterior loop and multiloop recursions with coaxial and dangle stacking, we make use of three new elementary recursion entities:

- *Coaxial stacking state:* two adjacent terminal base pairs that are coaxially stacked. Hence, a coaxial stacking state involves exactly two terminal base pairs.
- *Dangle stacking state:* zero, one, or two unpaired nucleotides dangle stacking on an adjacent terminal base pair. Hence, a dangle stacking state involves exactly one terminal base pair.
- *Stacking state:* a coaxial stacking state or a dangle stacking state (two adjacent terminal base pairs that are coaxially stacked or zero, one, or two unpaired nucleotides

dangle stacking on an adjacent terminal base pair). Hence, a stacking state involves either two or one terminal base pairs.

For example, a recursion might contain exactly one stacking state, a 3′-most stacking state, or one or more stacking states. Note that a terminal base pair without coaxial and dangle stacking corresponds to the subset of a dangle stacking state where there are zero nucleotides dangle stacking, so the complex ensemble without coaxial and dangle stacking is a subset of the complex ensemble with coaxial and dangle stacking.

To assist with examining the recursions with coaxial and dangle stacking, the intuition behind the name chosen for each recursion, the nature of the ensemble treated by each recursion, and the dependencies between the different recursions is summarized in Figure S13. To limit proliferation of new names and facilitate comparison to the non-stacking recursions of Section S2.3 (that treat complex ensemble $\overline{\Gamma}$ without coaxial and dangle stacking), we re-use the names of the non-stacking recursions but with updated recursion diagrams and recursion equations. Additionally, we introduce new recursions as needed to treat the coaxial and dangle stacking states in ensemble $\overline{\Gamma}^{||}$.

A recursion $R^a(i, j, \phi)$ operates on subsequence $[i : j]$ to calculate element i, j for either the unconstrained ensemble $a = \varnothing$ or for one of several constrained ensembles $a \in \{s, cd, b, n, x, ms, mcs, mc, md, m\}$. Briefly, $R^{\varnothing}(i, j, \phi)$ treats the unconstrained ensemble in an exterior loop context where i and j may or may not be paired. $R^s(i, j, \phi)$ serves as an efficiency wrapper over the 3′-most stacking state in an exterior loop context to reduce the time complexity from $O(N^4)$ to $O(N^3)$. $R^{cd}(i, j, \phi)$ treats a single stacking state (a coaxial stacking state or a dangle stacking state) in an exterior loop context. $R^b(i, j, \phi)$ treats the constrained ensemble where i and j form base pair $i \cdot j$ in the context of any loop type. $R^x(i, j, \phi)$ treats extensible interior loops to achieve $O(N^3)$ time complexity. $R^{ms}(i, j, \phi)$ serves as an efficiency wrapper over the 3′-most stacking state in a multiloop context (analogous to R^s in an exterior loop context) to reduce the time complexity from $O(N^4)$ to $O(N^3)$. $R^{mcs}(i, j, \phi)$ serves as an efficiency wrapper over the 3′-most coaxial stacking state in a multiloop context to reduce the time complexity from $O(N^4)$ to $O(N^3)$. $R^{mc}(i, j, \phi)$ treats a single coaxial stacking state in a multiloop context. $R^{md}(i, j, \phi)$ treats a single dangle stacking state in a multiloop context. $R^m(i, j, \phi)$ treats one or more remaining stacking states in a multiloop context.

When combined, R^{mc} and R^{md} constitute the multiloop equivalent to R^{cd} in an exterior loop context; they are kept separate to allow proper treatment of a multiloop edge case. In the exterior loop context, the efficiency wrapper R^s wraps R^{cd} to treat coaxial and dangle

Recursion	Naming intuition	Constraint	Context
$\varnothing$	unconstrained	none	exterior loop
s	**s**um	efficiency wrapper for 3′-most stacking state	exterior loop
cd	**c**oaxial and **d**angle	one stacking state (coaxial or dangle stacking state)	exterior loop
b	**b**ase-paired	base pair between 5′-most and 3′-most bases of subsequence	any loop
n	**n**ick	nick between strands	exterior loop
x	e**x**tensible	extensible interior loop	interior loop
ms	**m**ultiloop **s**um	efficiency wrapper for 3′-most stacking state	multiloop
mcs	**m**ultiloop **c**oaxial **s**um	efficiency wrapper for 3′-most coaxial stacking state	multiloop
mc	**m**ultiloop **c**oaxial	one coaxial stacking state	multiloop
md	**m**ultiloop **d**angle	one dangle stacking state	multiloop
m	**m**ultiloop	one or more remaining stacking states	multiloop

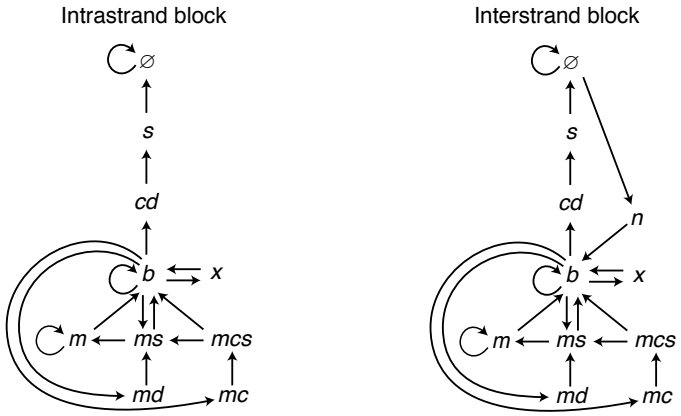


Figure S13: Nomenclature and connectivity for recursions with coaxial and dangle stacking. Top: Nomenclature. Bottom: Dependencies between different recursion types for elements within an intrastrand block (left) or an interstrand block (right).

stacking simultaneously. In the multiloop context, because of the edge case, the efficiency wrapper R^{mcs} wraps R^{mc} (to treat coaxial stacking alone) and then the efficiency wrapper R^{ms} incorporates R^{mcs} in addition to wrapping R^{md} (to treat dangle stacking alone). Hence, the efficiency wrapper R^{ms} (treating both coaxial and dangle stacking) is the multiloop equivalent to R^{s} in an exterior loop context.

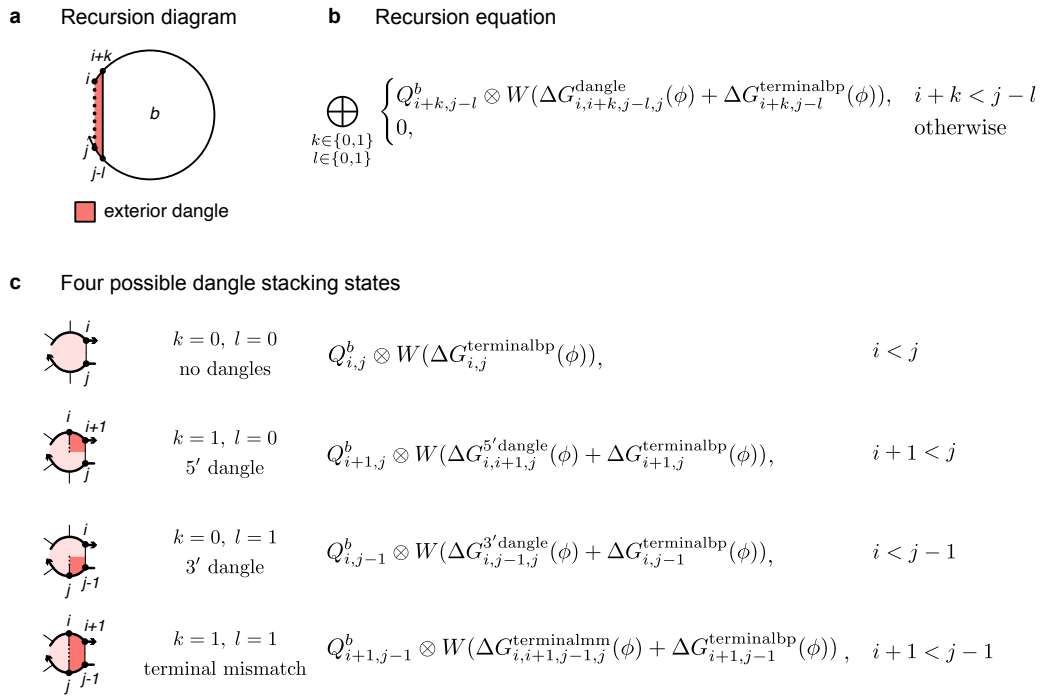
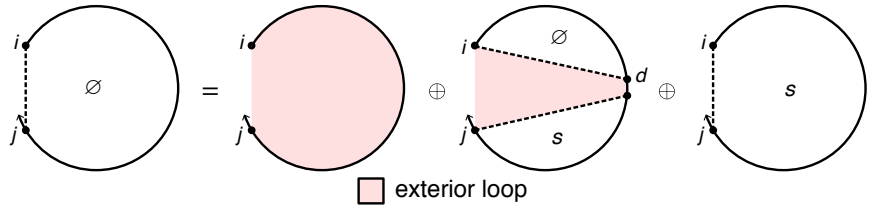


Figure S14: Summation over individual dangle states. (a) Example recursion diagram taken from the definition of $R^{cd}_{\text{INTRA}}(i,j,\phi)$. Note that the considered base pair is always between bases $i+k$ and $j-l$. (b) Equivalent recursion expression which specifies the specific free energy parameter contributions. (c) Decomposition of the sum in (b) into terms from each of 4 specific dangle states.

S2.6.1 Summation over dangle stacking states

Recursions that incorporate dangle stacking use a standardized approach to sum (using the $\oplus$ operator) over the subensemble of dangle stacking states on an adjacent terminal base pair. An example sum is depicted in the recursion diagram of Figure S14a by the dotted line between unpaired bases adjacent to a solid line denoting paired bases. The shading indicates explicit incorporation of a dangle free energy $\Delta G^{\text{dangle}}_{i,i+k,j-l,j}(\phi)$ (different for each dangle stacking state in the subensemble) and a terminal base pair free energy $\Delta G^{\text{terminalbp}}_{i+k,j-l}(\phi)$ (dependent on the sequence of base pair $i+k \cdot j-l$). The corresponding recursion equation of Figure S14b uses the indices $k \in \{0,1\}$ and $l \in \{0,1\}$ to sum over the four dangle stacking states, which are illustrated in Figure S14c. For $k=l=0$, there are no unpaired bases dangle stacking on terminal base pair $i \cdot j$. For $k=1,\ l=0$, there is a 5′ dangle stack on terminal base pair $i \cdot j$. For $k=0,\ l=1$, there is a 3′ dangle stack on terminal base pair $i \cdot j$. For $k=l=1$, there are both 5′ and 3′ dangle stacks on terminal base pair $i \cdot j$; this stacking state is referred to as a *terminal mismatch*. For clarity, recursion equations incorporate the generic dangle free energy function $\Delta G^{\text{dangle}}_{i,i+k,j-l,j}(\phi)$ which returns



$$R^{\varnothing}_{\text{INTRA}}(i,j,\phi) \equiv \mathbb{1} \oplus \begin{cases} Q^{s}_{i,j} \oplus \text{DOT}\left(Q^{\varnothing}_{i,\overline{d}}, Q^{s}_{\overline{d}+1,j}\right), & j-i>4 \\ Q^{s}_{i,j}, & j-i=4 \\ \mathbb{0}, & \text{otherwise} \end{cases} \tag{S56}$$

where $\overline{d} \equiv [i : j-5]$.

Figure S15: $R^{\varnothing}_{\text{INTRA}}$ recursion with coaxial and dangle stacking. Top: recursion diagram. Bottom: recursion equation.

the appropriate free energy for each of the four stacking states:

$$\Delta G^{\text{dangle}}_{i,i+k,j-l,j}(\phi) = \begin{cases} 0 & k=0,\ l=0, & \text{no dangles} \\ \Delta G^{5'\text{dangle}}_{i,i+1,j}(\phi) & k=1,\ l=0, & 5' \text{ dangle} \\ \Delta G^{3'\text{dangle}}_{i\,j-1,j}(\phi) & k=0,\ l=1, & 3' \text{ dangle} \\ \Delta G^{\text{terminalmm}}_{i,i+1,j-1,j}(\phi) & k=1,\ l=1, & \text{terminal mismatch.} \end{cases} \tag{S54}$$

Terminal mismatch free energies $\Delta G^{\text{terminalmm}}_{i,i+1,j-1,j}(\phi)$ have been published for RNA [14, 20] and are included in the `rna95` and `rna06` parameter sets. However, terminal mismatch parameters for DNA are not public [19]. As a result, the `dna04` parameter set assigns the terminal mismatch free energy to be the sum of the published 5′ and 3′ dangle free energies[18, 19]:

$$\Delta G^{\text{terminalmm}}_{i,i+1,j-1,j}(\phi) \equiv \Delta G^{5'\text{ dangle}}_{i,i+1,j}(\phi) + \Delta G^{3'\text{ dangle}}_{i\,j-1,j}(\phi). \tag{S55}$$

S2.6.2 Intrastrand dynamic programming recursions with coaxial and dangle stacking

Here, we consider recursions for calculating the entries in a triangular intrastrand block with coaxial and dangle stacking. By definition, there are no nicks between strands in intrastrand recursions since intrastrand blocks involve base-pairing within a single strand.

$R^{\varnothing}_{\text{INTRA}}$ recursion with coaxial and dangle stacking. We begin with the recursion $R^{\varnothing}_{\text{INTRA}}(i, j, \phi)$ with the diagram and equation shown in Figure S24. $R^{\varnothing}_{\text{INTRA}}(i, j, \phi)$ operates on the unconstrained ensemble for subsequence $[i, j]$ in an exterior loop context where i and j may or may not be paired (depicted with a dashed line between i and j in the recursion diagram). This recursion distinguishes two cases that are combined using $\oplus$ in the recursion equation:

- *No stacking states:* the empty case in an exterior loop context where there are no stacking states in subsequence $[i, j]$ (depicted by the absence of a straight solid line in the recursion diagram). The exterior loop shading in the recursion diagram represents the recursion energy $\Delta G^{\text{exterior}}_{i,j}(\phi) = 0$ corresponding to the zero reference state for an exterior loop with no coaxial stacking states or dangle stacking states (and hence, no terminal base pairs). The corresponding contribution to the recursion equation is $W(0) = \mathbb{1}$.

- *At least one stacking state:* the non-empty case in an exterior loop context where there is at least one stacking state (i.e., two adjacent terminal base pairs coaxially stacking, or zero, one, or two unpaired nucleotides dangle stacking on an adjacent terminal base pair) in subsequence $[i, j]$. The $3'$-most stacking state begins at $d + 1$ and ends in the interval $[d + 2, j]$ (depicted using a dashed line in the recursion diagram). The contributions for subsequence $[d + 1, j]$ are incorporated using a $Q^{s}_{d+1,j}$ element. Contributions for the remaining subsequence $[i, d]$ are incorporated by a $Q^{\varnothing}_{i,d}$ element. The shading denotes the recursion energy 0 corresponding to the zero reference state in an exterior loop context. The edge case where the index $d+1 = i$ is displayed explicitly to indicate that no $Q^{\varnothing}$ element is accessed in this case. The index limits in the recursion equation reflect the fact that steric effects prevent a hairpin loop with fewer than 3 unpaired nucleotides (hence, $i \cdot j$ cannot form if $j - i < 4$).

Note that using the DOT notation (Algorithm S1) and index range notation (S36) to denote vector operations, we have the equivalence:

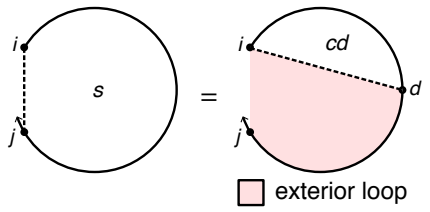
$$\text{DOT}\left(Q^{\varnothing}_{i,\overline{d}}, Q^{s}_{\overline{d}+1,j}\right) \equiv \sum_{d=i}^{j-5} Q^{\varnothing}_{i,d} \otimes Q^{s}_{d+1,j}, \quad j - i > 4.$$

$$\text{where } \overline{d} \equiv [i : j - 5].$$

We can also recognize that in terms of matrix elements, the dot product

$$\text{DOT}\left(Q^{\varnothing}_{i,\overline{d}}, Q^{s}_{\overline{d}+1,j}\right) \tag{S57}$$

is between the element range $\overline{d}$ of row i (depicted as brown elements in Figure S1b) and the element range $\overline{d}$+1 of column j (gray elements), yielding element i, j (purple element).



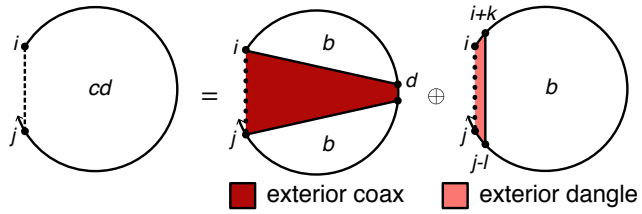
$$R^{s}_{\text{INTRA}}(i,j,\phi) \equiv \begin{cases} \text{DOT}\left(Q^{cd}_{i,\overline{d}}\right), & j-i \geq 4 \\ \mathbb{0}, & \text{otherwise} \end{cases}$$

$$\text{where } \overline{d} \equiv [i+4:j].$$

Figure S16: R^{s}_{INTRA} recursion with coaxial and dangle stacking. Top: recursion diagram. Bottom: recursion equation.

R^{s}_{INTRA} recursion with coaxial and dangle stacking. The $R^{\varnothing}_{\text{INTRA}}(i,j,\phi)$ recursion references Q^{s} elements that are computed using the R^{s}_{INTRA} recursion displayed in Figure S16. $R^{s}_{\text{INTRA}}(i,j,\phi)$ operates on a conditional ensemble for subsequence $[i,j]$ in an exterior loop context containing one stacking state starting at i and ending in the interval $[i+1,j]$ (depicted as a dashed line between i and j). The contribution for the stacking state in subsequence $[i,d]$ is incorporated using a $Q^{cd}_{i,d}$ element. Shading denotes no recursion energy as stacking energies and terminal base pair penalties are handled in R^{cd}. The index limits in the recursion equation reflect the steric requirement that there be at least 3 intervening bases between i and d (because the R^{cd} recursion incorporates a minimum of one terminal base pair in subsequence $[i:d]$). Note that the R^{s} recursion serves as an efficiency wrapper of the R^{cd} recursion (here, representing the 3′-most stacking state in an exterior loop context) to reduce the time complexity of the $R^{\varnothing}$ recursion from $O(N^4)$ to $O(N^3)$. This time complexity reduction is achieved by defining the 3′-most stacking state using R^{cd} within the R^{s} efficiency wrapper rather than directly using the R^{cd} recursion within the $R^{\varnothing}$ recursion, so as to avoid introducing a fourth independent index into the $R^{\varnothing}$ recursion.

R^{cd}_{INTRA} recursion with coaxial and dangle stacking. The $R^{s}_{\text{INTRA}}(i,j,\phi)$ recursion references Q^{cd} elements that are computed using the R^{cd}_{INTRA} recursion displayed in Figure S17.



$$R^{cd}_{\text{INTRA}}(i,j,\phi) \equiv \begin{cases} \text{DOT}\Big(Q^{b}_{i,\overline{d}}, Q^{b}_{\overline{d}+1,j}, \\ W(\Delta G^{\text{coax}}_{i,\overline{d},j}(\phi) + \Delta G^{\text{terminalbp}}_{i,\overline{d}}(\phi) + \Delta G^{\text{terminalbp}}_{\overline{d}+1,j}(\phi))\Big), & j-i \geq 9 \\ \mathbb{0}, & \text{otherwise} \end{cases}$$

$$\oplus \bigoplus_{\substack{k\in\{0,1\}\\ l\in\{0,1\}}} \begin{cases} Q^{b}_{i+k,j-l} \otimes \\ \quad W(\Delta G^{\text{dangle}}_{i,i+k,j-l,j}(\phi) + \Delta G^{\text{terminalbp}}_{i+k,j-l}(\phi)), & (j-l)-(i+k) \geq 4 \\ \mathbb{0}, & \text{otherwise} \end{cases}$$

where $\overline{d} \equiv [i+4 : j-5]$

Figure S17: R^{cd}_{INTRA} recursion with coaxial and dangle stacking. Top: recursion diagram. Bottom: recursion equation.

$R^{cd}_{\text{INTRA}}(i,j,\phi)$ treats a single stacking state in an exterior loop context, corresponding to either of two cases that are combined using $\oplus$ in the recursion equation:

- *Coaxial stacking state:* two adjacent terminal base pairs ($i \cdot d$ and $d+1 \cdot j$) coaxially stack on each other. The contributions of subsequences $[i,d]$ and $[d+1,j]$ are incorporated using $Q^{b}_{i,d}$ and $Q^{b}_{d+1,j}$ elements. Shading corresponds to two kinds of recursion energy: 1) the sequence-dependent penalties for two terminal base pairs in an exterior loop context, $\Delta G^{\text{terminalbp}}_{i,d}(\phi)$ and $\Delta G^{\text{terminalbp}}_{d+1,j}(\phi)$ (dependent on the sequence of base pairs $i \cdot d$ and $d+1 \cdot j$), 2) the sequence-dependent coaxial stacking free energy $\Delta G^{\text{coax}}_{i,d,j}(\phi)$ (dependent on the sequences of base pairs $i \cdot d$ and $d+1 \cdot j$). Note that $\Delta G^{\text{coax}}_{i,d,j}(\phi)$ requires only 3 indices because $d+1$ is implied by d. The index limits in the recursion equation reflect the steric requirement that there be at least 3 intervening bases between i and d and at least 3 intervening bases between $d+1$ and j.

- *Dangle stacking state:* zero, one, or two unpaired nucleotides dangle stack on an adjacent terminal base pair ($i+k \cdot j-l$). The recursion diagram summarizes four

dangle stacking states (depicted as a dotted line between i and j) corresponding to no dangles, 5′ dangle, 3′ dangle, or terminal mismatch (see Figure S14 for details). The contribution of subsequence $[i+k, j-l]$ is incorporated using $Q^b_{i+k,j-l}$ element. Shading corresponds to two recursion energies: 1) the sequence-dependent penalty for a terminal base pair in an exterior loop context, $\Delta G^{\text{terminalbp}}_{i+k,j-l}(\phi)$ (dependent on the sequence of base pair $i + k \cdot j - l$), 2) the sequence-dependent dangle stacking free energy $\Delta G^{\text{dangle}}_{i,i+k,j-l,j}(\phi)$ which takes on one of four values corresponding to the four dangle stacking states (see Figure S14). The index limits in the recursion equation reflect the steric requirement that there be at least 3 intervening bases between $i + k$ and $j - l$.

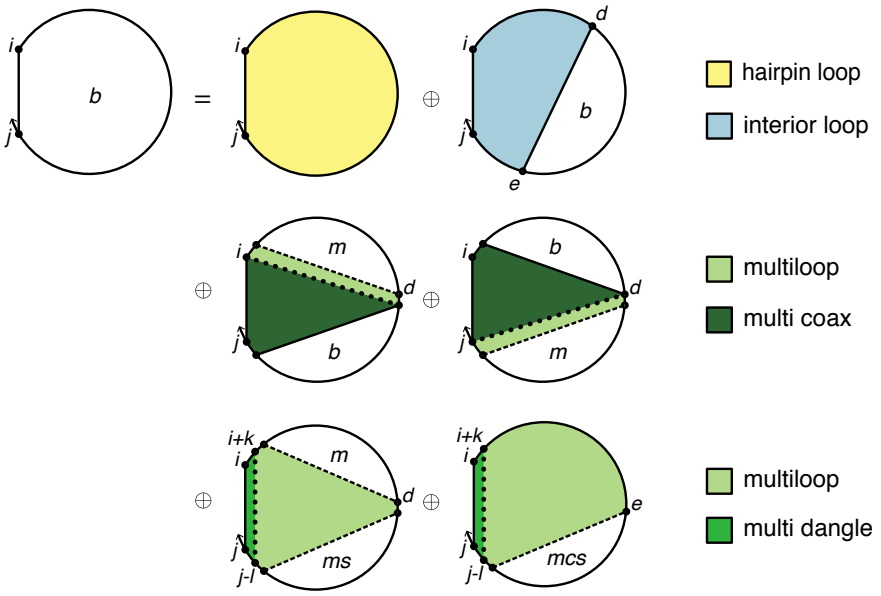
R^b_{INTRA} recursion with coaxial and dangle stacking. The $R^{cd}_{\text{INTRA}}(i, j, \phi)$ recursion references Q^b elements that are computed using the R^b_{INTRA} recursion displayed in Figure S18. $R^b_{\text{INTRA}}(i, j, \phi)$ operates on a conditional ensemble for subsequence $[i, j]$ with i and j base paired to each other (depicted with a solid line between i and j). The recursion distinguishes four cases that are combined using $\oplus$ in the recursion equation:

- *Hairpin loop:* the hairpin loop closed by the single base pair $i \cdot j$ (depicted by a straight solid line). The recursion incorporates the recursion energy $\Delta G^{\text{hairpin}}_{i,j}(\phi)$. This treatment of hairpin loops is the same as for the non-stacking recursions. The index limits in the recursion equation reflect the fact that steric constraints prevent a hairpin loop with fewer than 3 unpaired nucleotides (hence, $i \cdot j$ cannot form if $j - i < 4$).

- *Interior loop:* the interior loop closed by the two terminal base pairs $i \cdot j$ and $d \cdot e$ (depicted by straight solid lines). Calculation of the interior loop contributions using an $O(N^4)$ or $O(N^3)$ version of the InteriorIntra recursion is described in Section S2.4. This treatment of interior loops is the same as for the non-stacking recursions. The index limits in the recursion equation reflect the fact that steric effects prevent an interior loop with $j - i < 6$ due to the steric requirement that there be at least 3 intervening bases between d and e.

- *Multiloop with coaxial stacking on terminal base pair $j \cdot i$:* the multiloop closed by three or more terminal base pairs with coaxial stacking on base pair $j \cdot i$. This case corresponds to the two recursion diagrams on the second row of Figure S18 and is treated by the subroutine MultiCoaxIntra (recursion equation S58). The recursion

on the left treats the case where terminal base pair $j \cdot i$ forms a coaxial stack with adjacent terminal base pair $d+1 \cdot j-1$, depicted as a dotted straight line between i and $d+1$. The contribution of subsequence $[i+1, d]$ is incorporated by element $Q^{b}_{i+1,d}$. The contributions of one or more remaining stacking states in subsequence $[i+1, d]$ are incorporated by element $Q^{m}_{i+1,d}$. The pale green shading corresponds to three multiloop recursion energies: 1) the penalty for formation of a multiloop $\Delta G^{\text{multi}}_{\text{init}}$, 2) the sequence-independent penalties for two terminal base pairs in a multiloop, $\Delta G^{\text{multi}}_{\text{bp}}$



$$R^{b}_{\text{INTRA}}(i,j,\phi) \equiv \begin{cases} C_1, & \textsc{Complementary}(\phi_i, \phi_j) \\ \mathbb{0}, & \text{otherwise} \end{cases}$$

$$\text{where } C_1 \equiv \begin{cases} W(\Delta G^{\text{hairpin}}_{i,j}(\phi)), & j-i \geq 4 \\ \mathbb{0}, & \text{otherwise} \end{cases}$$

$$\oplus \begin{cases} \textsc{InteriorIntra}(i,j,\phi), & j-1 \geq 6 \\ \mathbb{0}, & \text{otherwise} \end{cases}$$

$$\oplus \begin{cases} \textsc{MultiCoaxIntra}(i,j,\phi) \oplus \textsc{MultiDangleIntra}(i,j,\phi), & j-i \geq 11 \\ \mathbb{0}, & \text{otherwise} \end{cases}$$

Figure S18: R^{b}_{INTRA} recursion with coaxial and dangle stacking. Top: recursion diagram. Bottom: recursion equation.

(corresponding to base pairs $d+1 \cdot j-1$ and $j \cdot i$), 3) the sequence-dependent penalties for two terminal base pairs in a multiloop context, $\Delta G^{\text{terminalbp}}_{d+1,j-1}(\phi)$ and $\Delta G^{\text{terminalbp}}_{j,i}(\phi)$ (note that the indices are ordered j then i to reflect 5′ to 3′ from the perspective of the multiloop). The dark green shading corresponds to the sequence-dependent coaxial stacking recursion energy $\Delta G^{\text{coax}}_{d+1,j-1,i}(\phi)$ (dependent on the sequences of base pairs $d+1 \cdot j-1$ and $j \cdot i$). Note that $\Delta G^{\text{coax}}_{d+1,j-1,i}(\phi)$ requires only 3 indices because j is implied by $j-1$. The recursion on the right treats the analogous case where terminal base pair $j \cdot i$ forms a coaxial stack with adjacent terminal base pair $i+1 \cdot d$. The index limits in the recursion equation reflect the fact that steric effects prevent a multiloop with $j-i<11$ due to the steric requirement that there be at least 3 intervening bases between $i+1$ and d (which must contain one or more stacking states and hence one or more terminal base pairs) and at least 3 intervening bases between $d+1$ and $j-1$.

- *Multiloop with dangle stacking on terminal base pair $j \cdot i$:* the multiloop closed by three or more terminal base pairs with dangle stacking on terminal base pair $j \cdot i$. This case corresponds to the two recursion diagrams on the third row of Figure S18 and is treated by the subroutine MultiDangleIntra (recursion equation S59).

  - *Base case with two or more additional stacking states.* The recursion on the left treats the case where there is a dangle stacking state involving the terminal base pair $j \cdot i$ (depicted as a dotted straight line between $i+k$ and $j-l$) and a 3′-most coaxial stacking state in subsequence $[d+1, j-l-1]$ (depicted as a dashed line between $d+1$ and $j-l-1$). The contributions for subsequence $[d+1, j-l-1]$ are incorporated using a $Q^{ms}_{d+1,j-l-1}$ element. The pale green shading corresponds to four multiloop recursion energies: 1) the penalty for formation of a multiloop $\Delta G^{\text{multi}}_{\text{init}}$, 2) the sequence-independent penalty for one terminal base pair in a multiloop, $\Delta G^{\text{multi}}_{\text{bp}}$ (corresponding to base pair $j \cdot i$), 3) the penalty per unpaired nucleotide in a multiloop $\Delta G^{\text{multi}}_{\text{nt}}$ (a total of $k+l$ dangling nucleotides; as a result this term is zeroed out when $k=l=0$). 4) the sequence-dependent penalty for a terminal base pair in a multiloop context, $\Delta G^{\text{terminalbp}}_{j,i}(\phi)$ (note that the indices are ordered j then i to reflect 5′ to 3′ from the perspective of the multiloop). The medium green shading corresponds to the sequence-dependent dangle stacking recursion energy $\Delta G^{\text{dangle}}_{j-l,j,i,i+k}(\phi)$. Note that $k,l \in \{0,1\}$ determine whether unpaired nucleotides dangle stack on the adjacent terminal base pair $j \cdot i$ in a multiloop context. The situation is analogous to that in an exterior loop context with $R^{cd}_{\text{Intra}}(i,j,\phi)$ (as detailed in Figure S14) with the only difference being that in the exterior loop context, $i+k$ and $j-l$

index the paired bases and in the multiloop context $i + k$ and $j - l$ index the unpaired bases. The index limits in the recursion equation reflect the fact that steric effects prevent a multiloop with $(j - l) - (i + k) < 11$ due to the steric requirement that there be at least 3 intervening bases between $i + k + 1$ and d (which must contain one or more stacking states and hence one or more terminal base pairs) and at least 3 intervening bases between $d + 1$ and $j - l - 1$ (which must contain a 3′-most stacking state and hence one or two terminal base pairs).

– *Edge case with one additional coaxial stacking state.* The recursion on the right treats the case where there is a dangle stacking state involving the terminal base pair $j \cdot i$ (depicted as a dotted straight line between $i + k$ and $j - l$) and a single coaxial stacking state in subsequence $[e, j - l - 1]$ (depicted as a dashed line between e and $j - l - 1$). The contributions for subsequence $[e, j - l - 1]$ are incorporated using a $Q^{mcs}_{e,j-l-1}$ element. The pale green shading corresponds to four multiloop recursion energies: 1) the penalty for formation of a multiloop $\Delta G^{\text{multi}}_{\text{init}}$, 2) the sequence-independent penalty for one terminal base pair in a multiloop, $\Delta G^{\text{multi}}_{\text{bp}}$ (corresponding to base pair $j \cdot i$), 3) the penalty per unpaired nucleotide in a multiloop $\Delta G^{\text{multi}}_{\text{nt}}$ ($k + l$ dangling nucleotides plus the unpaired nucleotides $k+l+1, \ldots, e-1$; as a result this term is zeroed out when $k = l = 0$ and $e = i + 1$). 4) the sequence-dependent penalty for a terminal base pair in a multiloop context, $\Delta G^{\text{terminalbp}}_{j,i}(\phi)$ (note that the indices are ordered j then i to reflect 5′ to 3′ from the perspective of the multiloop). The medium green shading corresponds to the sequence-dependent dangle stacking recursion energy $\Delta G^{\text{dangle}}_{j-l,j,i,i+k}(\phi)$. The index limits in the recursion equation reflect the fact that steric effects prevent a multiloop with $(j - l) - (i + k) < 11$ due to the steric requirement that there be at least 8 intervening bases between e and $j - l - 1$ (which must contain a coaxial stacking state and hence two adjacent terminal base pairs). Note that this edge case covers the scenario where there are exactly three terminal base pairs and the two terminal base pairs that are not $j \cdot i$ are coaxially stacked. That situation is not covered by the base case because for that recursion, a multiloop with 3 terminal base pairs would have one terminal base pair in the Q^{ms} element, and one terminal base pair in the Q^{m} element (hence, those two terminal base pairs cannot coaxially stack since they are in different recursions).

$$\begin{aligned}
&\textsc{MultiCoaxIntra}(i,j,\phi) \equiv C_1 \otimes W(\Delta G^{\text{multi}}_{\text{init}} + 2\Delta G^{\text{multi}}_{\text{bp}} + \Delta G^{\text{terminalbp}}_{j,i}(\phi)) \\
&\text{where } C_1 \equiv \textsc{dot}\left(Q^{b}_{i+1,\overline{d}}, Q^{m}_{\overline{d}+1,j-1}, W(\Delta G^{\text{coax}}_{j,i,\overline{d}}(\phi) + \Delta G^{\text{terminalbp}}_{i+1,\overline{d}}(\phi))\right) \\
&\qquad\qquad \oplus \textsc{dot}\left(Q^{m}_{i+1,\overline{d}}, Q^{b}_{\overline{d}+1,j-1}, W(\Delta G^{\text{coax}}_{\overline{d}+1,j-1,i}(\phi) + \Delta G^{\text{terminalbp}}_{\overline{d}+1,j-1}(\phi))\right) \\
&\overline{d} \equiv [i+5 : j-6]
\end{aligned} \tag{S58}$$

$$\textsc{MultiDangleIntra}(i,j,\phi) \equiv \bigoplus_{\substack{k\in\{0,1\}\\ l\in\{0,1\}}} \begin{cases} C_1 \oplus C_2, & (j-l)-(i+k) \geq 11 \\ \mathbb{0}, & \text{otherwise} \end{cases}$$

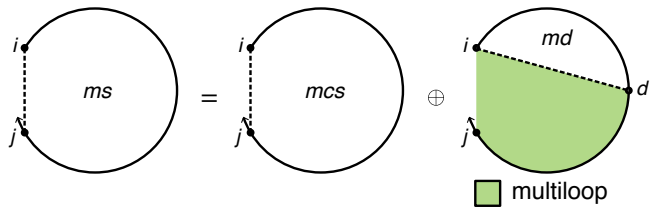
$$\begin{aligned}
&\text{where } C_1 \equiv \textsc{dot}\left(Q^{m}_{i+k+1,\overline{d}}, Q^{ms}_{\overline{d}+1,j-l-1}\right) \\
&\qquad \otimes W(\Delta G^{\text{dangle}}_{j-l,j,i,i+k}(\phi) + \Delta G^{\text{multi}}_{\text{init}} + \Delta G^{\text{multi}}_{\text{bp}} + (k+l)\Delta G^{\text{multi}}_{\text{nt}} + \Delta G^{\text{terminalbp}}_{j,i}(\phi)) \\
&C_2 \equiv \textsc{dot}\left(Q^{mcs}_{\overline{e},j-l-1}\right) \\
&\qquad \otimes W(\Delta G^{\text{dangle}}_{j-l,j,i,i+k}(\phi) + \Delta G^{\text{multi}}_{\text{init}} + \Delta G^{\text{multi}}_{\text{bp}} + \overline{n}_{\text{nt}}\Delta G^{\text{multi}}_{\text{nt}} + \Delta G^{\text{terminalbp}}_{j,i}(\phi)) \\
&\overline{d} \equiv [i+k+5 : j-l-6], \quad \overline{e} \equiv [i+k+1 : j-l-10], \quad \overline{n}_{\text{nt}} \equiv [k+l : j-i-11].
\end{aligned} \tag{S59}$$

$R^{ms}_{\textsc{Intra}}$ recursion with coaxial and dangle stacking. The $R^{b}_{\textsc{Intra}}(i,j,\phi)$ recursion references Q^{ms} elements that are computed using the $R^{ms}_{\textsc{Intra}}$ recursion shown in Figure S19. $R^{ms}_{\textsc{Intra}}(i,j,\phi)$ operates on a conditional ensemble for subsequence $[i,j]$ in a multiloop context containing one stacking state starting at i and ending in the interval $[i+1,j]$ (depicted as a dashed line between i and j). There are two cases that are combined using $\oplus$ in the recursion equation:

- *Coaxial stacking state:* The contribution for the coaxial stacking state in subsequence $[i,j]$ is calculated using a $Q^{mcs}_{i,j}$ element.
- *Dangle stacking state:* The contribution for the dangle stacking state in subsequence $[i,d]$ is incorporated using a $Q^{md}_{i,d}$ element. Shading corresponds to the recursion

energy penalty per unpaired nucleotide in a multiloop, $\Delta G_{\text{nt}}^{\text{multi}}$ (nucleotides $d+1, \ldots, j$ for a total of $j-d$ unpaired nucleotides; as a result, this term is zeroed out in the edge case where $d = j$). Note that in the dot product the range multiplying $\Delta G_{\text{nt}}^{\text{multi}}$ runs in reverse order because the number of unpaired nucleotides, $j-d$, decreases in size as d increases in size.

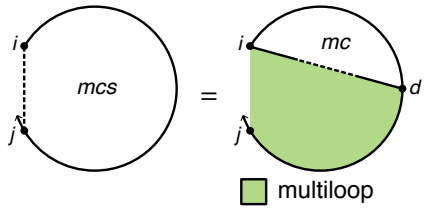
Note that R^{ms} directly incorporates the R^{mcs} recursion which serves as an efficiency wrapper of the R^{mc} recursion, and hence, R^{ms} is an efficiency wrapper of R^{mc} (the 3′-most coaxial stacking state in a multiloop context). Note also that R^{ms} is an efficiency wrapper of the R^{md} recursion (the 3′-most dangle stacking state in a multiloop context). Taken together R^{mc} and R^{md} represent the 3′-most stacking state in a multiloop context, analogous to R^{cd} representing the 3′-most stacking state (coaxial or dangle) in an exterior loop context. The reason that R^{mc} (coaxial stacking states) and R^{md} (dangle stacking states) are calculated and stored separately in a multiloop context is that coaxial-only information (stored in element Q^{mcs}) is needed for the previously described multiloop edge case (right recursion diagram in the third row of Figure S18). As a result, coaxial-only information is calculated using the efficiency wrapper R^{mcs} for use in that edge case, and then coaxial-only and dangle-only information are combined by the R^{ms} efficiency wrapper (which is fully analogous to the R^{s} efficiency wrapper in the exterior loop context). With this approach, the operations spent calculating coaxial stacking information for Q^{mcs} elements are not repeated when calculating both coaxial and dangle stacking for Q^{ms} elements.



$$R^{ms}_{\text{INTRA}}(i,j,\phi) \equiv Q^{mcs}_{i,j} \oplus \begin{cases} \text{DOT}\left(Q^{md}_{i,\overline{d}}, W(\overline{n}_{\text{nt}}\Delta G^{\text{multi}}_{\text{nt}})\right), & j-i \geq 4 \\ \mathbb{0}, & \text{otherwise} \end{cases}$$

$$\text{where } \overline{d} \equiv [i+4 : j], \qquad \overline{n}_{\text{nt}} \equiv [0 : j-i-4]^r$$

Figure S19: R^{ms}_{INTRA} recursion with coaxial and dangle stacking. Top: recursion diagram. Bottom: recursion equation.



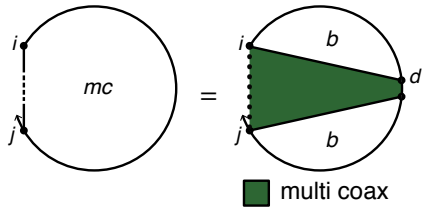
$$R^{mcs}_{\text{INTRA}}(i,j,\phi) \equiv \begin{cases} \text{DOT}\left(Q^{mc}_{i,\overline{d}}, W(\overline{n}_{\text{nt}}\Delta G^{\text{multi}}_{\text{nt}})\right), & j-i \geq 9 \\ \mathbb{0}, & \text{otherwise} \end{cases}$$

$$\text{where } \overline{d} \equiv [i+9:j], \qquad \overline{n}_{\text{nt}} \equiv [0:j-i-9]^r$$

Figure S20: R^{mcs}_{INTRA} recursion with coaxial and dangle stacking. Top: recursion diagram. Bottom: recursion equation.

R^{mcs}_{INTRA} recursion with coaxial and dangle stacking. The $R^{ms}_{\text{INTRA}}(i,j,\phi)$ recursion references Q^{mcs} elements that are computed using the R^{mcs}_{INTRA} recursion displayed in Figure S20. $R^{mcs}_{\text{INTRA}}(i,j,\phi)$ operates on a conditional ensemble for subsequence $[i,j]$ in a multiloop context containing one coaxial stacking state starting at i and ending in the interval $[i+1,j]$ (depicted as a dashed line between i and j). The contribution for the coaxial stacking state in subsequence $[i,d]$ is incorporated using a $Q^{mc}_{i,d}$ element. Shading denotes the penalty per unpaired nucleotide in a multiloop $\Delta G^{\text{multi}}_{\text{nt}}$ (the unpaired nucleotides $d+1,\ldots,j$; as a result this term is zeroed out when $d=j$). Note that in the dot product the range multiplying $\Delta G^{\text{multi}}_{\text{nt}}$ runs in reverse order because the number of unpaired nucleotides, $j-d$, decreases in size as d increases in size. Note that the R^{mcs} recursion serves as an efficiency wrapper of the R^{mc} recursion (here, representing the 3′-most coaxial stacking state in a multiloop context). The index limits in the recursion equation reflect the steric requirement that there be at least 8 intervening bases between i and d (because the R^{mc} recursion incorporates a coaxial stack involving two adjacent terminal base pairs such that i and d are paired to intervening adjacent bases).

R^{mc}_{INTRA} recursion with coaxial and dangle stacking. The $R^{mcs}_{\text{INTRA}}(i,j,\phi)$ recursion references Q^{mc} elements that are computed using the R^{mc}_{INTRA} recursion displayed in Figure S21. This recursion treats a single coaxial stacking state in a multiloop context (depicted as a straight line between i and j that is solid at both ends and dashed in the middle to indicate that i and j are both base-paired but not to each other). Two adjacent terminal base pairs ($i\cdot d$ and $d+1\cdot j$) coaxially stack on each other. The contributions of subsequences $[i,d]$ and



$$R^{mc}_{\text{INTRA}}(i,j,\phi) \equiv \begin{cases} C_1 \otimes W(2\Delta G^{\text{multi}}_{\text{bp}}), & j-i \geq 9 \\ \mathbb{0}, & \text{otherwise} \end{cases}$$

$$\text{where } C_1 \equiv \text{DOT}\left(Q^b_{i,\overline{d}}, Q^b_{\overline{d}+1,j}, W(\Delta G^{\text{coax}}_{i,\overline{d},j}(\phi) + \Delta G^{\text{terminalbp}}_{i,\overline{d}}(\phi) + \Delta G^{\text{terminalbp}}_{\overline{d}+1,j}(\phi))\right)$$

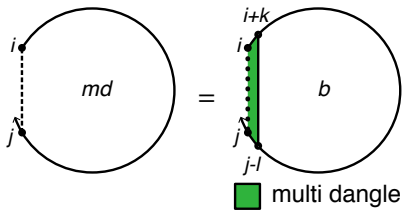
$$\overline{d} \equiv [i+4 : j-5]$$

Figure S21: R^{mc}_{INTRA} recursion with coaxial and dangle stacking. Top: recursion diagram. Bottom: recursion equation.

$[d+1, j]$ are incorporated using $Q^b_{i,d}$ and $Q^b_{d+1,j}$ elements. Shading corresponds to three kinds of recursion energy: 1) the sequence-independent penalties for two terminal base pairs in a multiloop, $\Delta G^{\text{multi}}_{\text{bp}}$ (corresponding to base pairs $i \cdot d$ and $d+1 \cdot j$), 2) the sequence-dependent penalties for two terminal base pairs in a multiloop context, $\Delta G^{\text{terminalbp}}_{i,d}(\phi)$ and $\Delta G^{\text{terminalbp}}_{d+1,j}(\phi)$ (dependent on the sequence of base pairs $i \cdot d$ and $d+1 \cdot j$), 3) the sequence-dependent coaxial stacking free energy $\Delta G^{\text{coax}}_{i,d,j}(\phi)$ (dependent on the sequences of base pairs $i \cdot d$ and $d+1 \cdot j$). Note that $\Delta G^{\text{coax}}_{i,d,j}(\phi)$ requires only 3 indices because $d+1$ is implied by d. The index limits in the recursion equation reflect the steric requirement that there be at least 3 intervening bases between i and d and at least 3 intervening bases between $d+1$ and j.

R^{md}_{INTRA} recursion with coaxial and dangle stacking. The $R^{ms}_{\text{INTRA}}(i,j,\phi)$ recursion references Q^{md} elements that are computed using the R^{md}_{INTRA} recursion displayed in Figure S22. This recursion treats a single dangle stacking state (depicted as a dashed line between i and j) in a multiloop context with either zero, one, or two unpaired nucleotides dangle stacking on an adjacent terminal base pair ($i+k \cdot j-l$). The recursion diagram represents these four alternative dangle stacking states corresponding to no dangles, 5′ dangle, 3′ dangle, or terminal mismatch (see Figure S14 for details). The contribution of subsequence $[i+k, j-l]$ is incorporated using a $Q^b_{i+k,j-l}$ element. Shading corresponds to four recursion energies: 1) the sequence-independent penalty for one terminal base pair in a multiloop,

$\Delta G_{\text{bp}}^{\text{multi}}$ (corresponding to base pair $j - l \cdot i + k$), 2) the penalty per unpaired nucleotide in a multiloop $\Delta G_{\text{nt}}^{\text{multi}}$ (a total of $k + l$ dangling nucleotides; as a result this term is zeroed out when $k = l = 0$). 3) the sequence-dependent penalty for a terminal base pair in a multiloop loop context, $\Delta G_{i+k,j-l}^{\text{terminalbp}}(\phi)$ (dependent on the sequence of base pair $i + k \cdot j - l$), 4) the sequence-dependent dangle stacking free energy $\Delta G_{i,i+k,j-l,j}^{\text{dangle}}(\phi)$ which takes on one of four values corresponding to the four dangle stacking states (see Figure S14). The index limits in the recursion equation reflect the steric requirement that there be at least 3 intervening bases between $i + k$ and $j - l$.



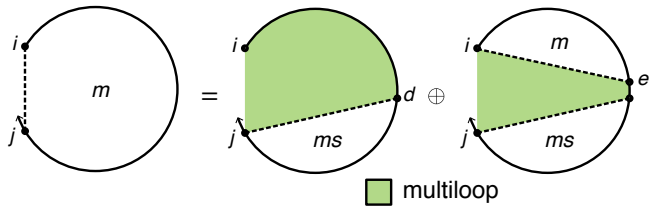
$$R_{\text{INTRA}}^{md}(i,j,\phi) \equiv \bigoplus_{\substack{k\in\{0,1\}\\ l\in\{0,1\}}} \begin{cases} C_1 & (j-l)-(i+k) \geq 4 \\ \mathbb{0}, & \text{otherwise} \end{cases}$$

$$\text{where } C_1 \equiv Q_{i+k,j-l}^{b} \otimes W(\Delta G_{i,i+k,j-l,j}^{\text{dangle}}(\phi) + \Delta G_{\text{bp}}^{\text{multi}} + (k+l)\Delta G_{\text{nt}}^{\text{multi}} + \Delta G_{i+k,j-l}^{\text{terminalbp}}(\phi))$$

Figure S22: R_{INTRA}^{md} recursion with coaxial and dangle stacking. Top: recursion diagram. Bottom: recursion equation.

R_{INTRA}^{m} recursion with coaxial and dangle stacking. The $R_{\text{INTRA}}^{b}(i,j,\phi)$ recursion also references Q^m elements that are computed using the R_{INTRA}^{m} recursion shown in Figure S23. $R_{\text{INTRA}}^{m}(i,j,\phi)$ operates on a conditional ensemble for subsequence $[i,j]$ in a multiloop context where i and j may or may not be paired (depicted with a dashed line between i and j in the recursion diagram) and where there is at least one stacking state. This recursion distinguishes two cases that are combined using $\oplus$ in the recursion equation:

- *One stacking state:* the case where there is exactly one stacking state in subsequence $[i,j]$ in a multiloop context. This stacking state starts at d and ends in the interval $[d+1,j]$ (depicted by a straight dashed line between d and j); the contribution of subsequence $[d,j]$ is incorporated by element $Q_{d,j}^{ms}$. Shading corresponds to the recursion energy, $\Delta G_{\text{nt}}^{\text{multi}}$, representing the penalty per unpaired nucleotide in a multiloop (nucleotides $i,\ldots,d-1$ for a total of $d-i$ unpaired nucleotides; as a



$$R^{m}_{\text{INTRA}}(i,j,\phi) \equiv \begin{cases} \text{DOT}\left(Q^{ms}_{\overline{d},j}), W(\overline{n}_{\text{nt}}\Delta G^{\text{multi}}_{\text{nt}})\right), & j-i \geq 4 \\ \mathbb{0}, & \text{otherwise} \end{cases}$$

$$\oplus \begin{cases} \text{DOT}\left(Q^{m}_{i,\overline{e}}, Q^{ms}_{\overline{e}+1,j}\right), & j-i \geq 9 \\ \mathbb{0}, & \text{otherwise} \end{cases}$$

$$\text{where } \overline{d} \equiv [i : j-4], \quad \overline{n}_{\text{nt}} \equiv [0 : j-i-4], \quad \overline{e} \equiv [i+4 : j-5]$$

Figure S23: R^{m}_{INTRA} recursion with coaxial and dangle stacking. Top: recursion diagram. Bottom: recursion equation.

result, this term is zeroed out in the edge case where $d = i$). The index limits in the recursion equation reflect the steric requirement that there be at least 3 intervening bases between d and j (because the R^{ms} incorporates a minimum of one terminal base pair in subsequence $[d : j]$).

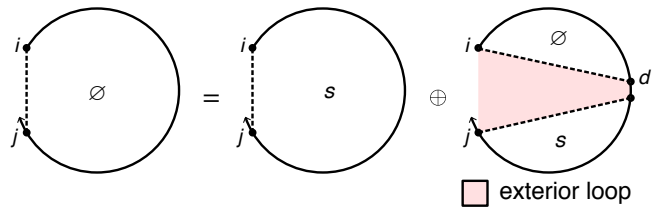
- *More than one stacking state:* the case where there are two or more stacking states in subsequence $[i, j]$ in a multiloop context. The 3′-most stacking state starts at $e+1$ and ends in the interval $[e+2, j]$ (depicted by a straight dashed line between $e+1$ and j); the contribution of subsequence $[e+1, j]$ is incorporated by element $Q^{ms}_{e+1,j}$. There are one or more additional stacking states in the interval $[i, e]$ (the straight dashed line denotes that i and e may or may not be paired); the contribution of subsequence $[i, e]$ is incorporated by element $Q^{m}_{i,e}$. The shading does not represent any recursion energies as all multiloop contributions are handled by other recursions: 1) there are no terminal base pairs in a multiloop context explicitly defined in this case, 2) there are no unpaired bases in a multiloop context explicitly defined in this case. The index limits in the recursion equation reflect the steric requirement that there be at least 3 intervening bases between i and e and at least 3 intervening bases between $e+1$ and j.

S2.6.3 Interstrand dynamic programming recursions with coaxial and dangle stacking

Here, we consider recursions for calculating the entries in a rectangular interstrand block with coaxial and dangle stacking. By definition, interstrand blocks involve 2 or more strands, and hence one or more nicks between strands. For a given interstrand block, η stores an array of nick indices between strands within the block, with each nick denoted by the index of the nucleotide following the nick. If $m \equiv \textsc{First}(\eta)$ and $n \equiv \textsc{Last}(\eta)$, then for subsequence $[i, j]$ corresponding to element i, j in the interstrand block, we have by definition $i < m$ (nucleotide i is on the first strand in the block) and $j \geq n$ (nucleotide j is on the last strand in the block).

$R^{\varnothing}_{\textsc{Inter}}$ recursion with coaxial and dangle stacking. We begin with $R^{\varnothing}_{\textsc{Inter}}(i, j, \phi)$ shown in Figure S24. $R^{\varnothing}_{\textsc{Inter}}(i, j, \phi)$ operates on the unconstrained ensemble for subsequence $[i, j]$ with i and j on different strands in an exterior loop context where i and j may or may not be paired (depicted with a dashed line between i and j in the recursion diagram). Unlike $R^{\varnothing}_{\textsc{Intra}}(i, j, \phi)$, there is no empty case because this would correspond to a disconnected structure (which is not in the ensemble) due to the presence of one or more nicks between i and j. Hence, the only case is *at least one stacking state:* the non-empty case in an exterior loop context where there is at least one stacking state (i.e., two adjacent terminal base pairs coaxially stacked or zero, one, or two unpaired nucleotides dangle stacking on an adjacent terminal base pair) in subsequence $[i, j]$. The $3'$-most stacking state begins at $d + 1$ and ends in the interval $[d + 2, j]$ (depicted using a dashed line in the recursion diagram). The contributions for subsequence $[d + 1, j]$ are incorporated using a $Q^{s}_{d+1,j}$ element. Contributions for the remaining subsequence $[i, d]$ are incorporated by a $Q^{\varnothing}_{i,d}$ element. The shading denotes the recursion energy 0 corresponding to the zero reference state in an exterior loop context. The edge case where the index $d + 1 = i$ is displayed explicitly to indicate that no $Q^{\varnothing}$ element is accessed in this case.

Because there are nicks involved in calculating the elements of interstrand blocks, care must be taken to ensure that no disconnected secondary structures are incorporated in the complex ensemble. For a given interstrand block with nick indices η, the function $\textsc{Valid}$ returns the set of valid vectorization ranges $\{\overline{d}_1, \overline{d}_2, \dots\}$, such that for each valid vectorization range, d and $d + 1$ are on the same strand (i.e., such that d and $d + 1$ do not take on values that would place a nick between them). As is evident from the recursion diagram of Figure S8, if d and $d + 1$ were to take on values that placed a nick between them, a disconnected structure would result. There is at most one valid vectorization range per strand, and there may be



$$R^{\varnothing}_{\text{INTER}}(i,j,\phi) \equiv Q^{s}_{i,j} \oplus \bigoplus_{\overline{d}\in \text{VALID}(i,\max(j-4,n),\eta)} \text{DOT}\left(Q^{\varnothing}_{i,\overline{d}}, Q^{s}_{\overline{d}+1,j}\right)$$

where $n = \text{LAST}(\eta)$

Figure S24: $R^{\varnothing}_{\text{INTER}}$ recursion with coaxial and dangle stacking. Top: recursion diagram. Bottom: recursion equation.

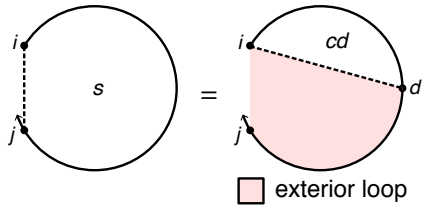
none for a strand or subsequence that is too short. For each valid vectorization range $\overline{d}$, the resulting dot product

$$\text{DOT}\left(Q^{\varnothing}_{i,\overline{d}}, Q^{s}_{\overline{d}+1,j}\right) \tag{S60}$$

is between the range $\overline{d}$ of row i (depicted as brown elements in Figure S1c) and the range $\overline{d}+1$ of column j (gray elements), yielding element i, j (purple element). Note that Figure S1c depicts two valid vectorization ranges (leading to two dot products that are summed to calculate the purple element); the gap of one element between the two vectorization ranges corresponds to exclusion of the value $d = 3$ which would have placed a nick between nucleotides d and $d + 1$ (note that $\eta = 4$ for this interstrand block).

Note that for calculating element i, j, the subsequence submitted to VALID ranges from i to $\max(j - 4, n)$, where $n \equiv \text{LAST}(\eta)$. This yields two cases:

- If $\max(j - 4, n) = j - 4$: there is no nick between nucleotide $j - 4$ and j (since $n \equiv \text{LAST}(\eta) < j - 4$), so there must be at least 3 intervening bases between $d + 1$ and j because steric effects prevent a hairpin loop with fewer than 3 unpaired nucleotides. In this case, each incorporated element $Q^{s}_{d+1,j}$ results from an $R^{s}_{\text{INTRA}}(d + 1, j, \phi)$ recursion for an intrastrand block.
- If $\max(j - 4, n) = n$: there is a nick between nucleotide $j - 4$ and j (since $n \geq j - 4$), so $d + 1$ can be as large as $n - 1$ and still pair to any nucleotide in subsequence $[n, j]$. In this case, each incorporated element $Q^{s}_{d+1,j}$ results from an $R^{s}_{\text{INTER}}(d + 1, j, \phi)$ recursion for an interstrand block.



$$R^{s}_{\text{INTER}}(i,j,\phi) \equiv \text{DOT}\left(Q^{cd}_{i,\overline{d}}\right)$$

$$\text{where } \overline{d} \equiv [\text{LAST}(\eta) : j]$$

Figure S25: R^{s}_{INTER} recursion with coaxial and dangle stacking. Top: recursion diagram. Bottom: recursion equation.

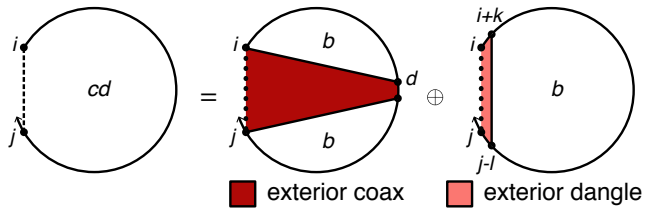
R^{s}_{INTER} recursion with coaxial and dangle stacking. The $R^{\varnothing}_{\text{INTRA}}(i,j,\phi)$ recursion references $Q^{s}_{d+1,j}$ elements that are computed using either the R^{s}_{INTRA} recursion of Figure S16 (if $d+1$ and j are on the same strand) or the R^{s}_{INTER} recursion of Figure S25 (if $d+1$ and j are on different strands). Recursion $R^{s}_{\text{INTER}}(i,j,\phi)$ operates on a conditional ensemble for subsequence $[i,j]$ with i and j on different strands in an exterior loop context containing one stacking state starting at i and ending in the interval $[i+1,j]$ (depicted as a dashed line between i and j). The contribution for the stacking state in subsequence $[i,d]$ is incorporated using a $Q^{cd}_{i,d}$ element. Shading denotes no recursion energy as stacking energies and terminal base pair penalties are handled in R^{cd}. The index d must always be on the last strand (i.e., $d \geq \text{LAST}(\eta)$) to ensure there are no strand breaks in the subsequence $[d,j]$, which would correspond to a disconnected structure. Note that the R^{s} recursion serves as an efficiency wrapper of the R^{cd} recursion (here, representing the 3′-most stacking state in an exterior loop context) to reduce the time complexity of the $R^{\varnothing}$ recursion from $O(N^4)$ to $O(N^3)$. This time complexity reduction is achieved by defining the 3′-most stacking state using R^{cd} within the R^{s} efficiency wrapper rather than directly using the R^{cd} recursion within the $R^{\varnothing}$ recursion, so as to avoid introducing a fourth independent index into the $R^{\varnothing}$ recursion.

R^{cd}_{INTER} recursion with coaxial and dangle stacking. The $R^{s}_{\text{INTER}}(i,j,\phi)$ recursion references Q^{cd} elements that are computed using either the R^{cd}_{INTRA} recursion of Figure S17 (if i and d are on the same strand) or the R^{cd}_{INTER} recursion of Figure S26 (if i and d are on different strands). Recursion $R^{cd}_{\text{INTER}}(i,j,\phi)$ treats a single stacking state in an exterior loop

context, corresponding to either of two cases that are combined using $\oplus$ in the recursion equation:

- *Coaxial stacking state:* two adjacent terminal base pairs ($i \cdot d$ and $d+1 \cdot j$) coaxially stack on each other. The contributions of subsequences $[i, d]$ and $[d+1, j]$ are incorporated using $Q^b_{i,d}$ and $Q^b_{d+1,j}$ elements. Shading corresponds to two kinds of recursion energy: 1) the sequence-dependent penalties for two terminal base pairs in an exterior loop context, $\Delta G^{\text{terminalbp}}_{i,d}(\phi)$ and $\Delta G^{\text{terminalbp}}_{d+1,j}(\phi)$ (dependent on the sequence of base pairs $i \cdot d$ and $d+1 \cdot j$), 2) the sequence-dependent coaxial stacking free energy $\Delta G^{\text{coax}}_{i,d,j}(\phi)$ (dependent on the sequences of base pairs $i \cdot d$ and $d+1 \cdot j$). Note that $\Delta G^{\text{coax}}_{i,d,j}(\phi)$ requires only 3 indices because $d+1$ is implied by d. To ensure that disconnected structures are excluded from the ensemble, the function VALID returns the set of valid vectorization ranges for which nucleotides d and $d+1$ are on the same strand (i.e., such that d and $d+1$ do not take on values that would place a nick between them).

- *Dangle stacking state:* zero, one, or two unpaired nucleotides dangle stack on an adjacent terminal base pair ($i+k \cdot j-l$). The recursion diagram summarizes four dangle stacking states (depicted as a dotted line between i and j) corresponding to no dangles, 5′ dangle, 3′ dangle, or terminal mismatch (see Figure S14 for details).



$$R^{cd}_{\text{INTER}}(i,j,\phi) \equiv \bigoplus_{\overline{d} \in \text{VALID}(i,j,\eta)} \text{DOT}\left(Q^b_{i,\overline{d}}, Q^b_{\overline{d}+1,j}, W(\Delta G^{\text{coax}}_{i,\overline{d},j}(\phi) + \Delta G^{\text{terminalbp}}_{i,\overline{d}}(\phi) + \Delta G^{\text{terminalbp}}_{\overline{d}+1,j}(\phi))\right)$$
$$\oplus \bigoplus_{\substack{k \in \{0,1\} \\ l \in \{0,1\}}} \begin{cases} Q^b_{i+k,j-l} \otimes W(\Delta G^{\text{dangle}}_{i,i+k,j-l,j}(\phi) + \Delta G^{\text{terminalbp}}_{i+k,j-l}(\phi)), & i+k < m \text{ and } j-l \geq n \\ \mathbb{0}, & \text{otherwise} \end{cases}$$

where $m = \text{FIRST}(\eta)$
$n = \text{LAST}(\eta)$

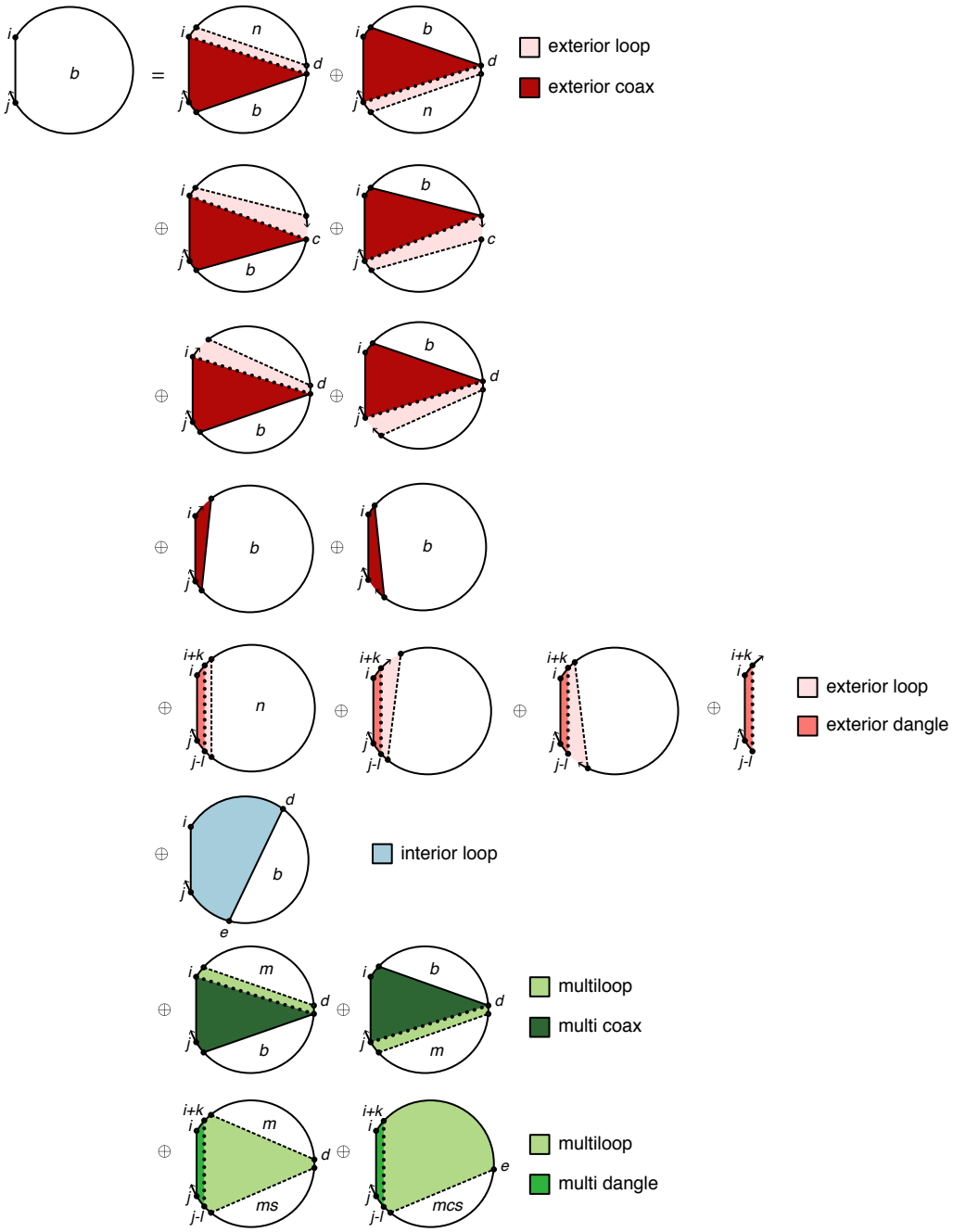
Figure S26: R^{cd}_{INTER} recursion with coaxial and dangle stacking. Top: recursion diagram. Bottom: recursion equation.

The contribution of subsequence $[i+k, j-l]$ is incorporated using $Q^b_{i+k,j-l}$ element. Shading corresponds to two recursion energies: 1) the sequence-dependent penalty for a terminal base pair in an exterior loop context, $\Delta G^{\text{terminalbp}}_{i+k,j-l}(\phi)$ (dependent on the sequence of base pair $i+k \cdot j-l$), 2) the sequence-dependent dangle stacking free energy $\Delta G^{\text{dangle}}_{i,i+k,j-l,j}(\phi)$ which takes on one of four values corresponding to the four dangle stacking states (see Figure S14). To ensure that disconnected structures are excluded from the ensemble, the index limits in the recursion equation prevent a nick between a dangling nucleotide and the base pair on which it stacks (i.e., no nick between i and $i+k$, and no nick between j and $j-l$).

R^b_{INTER} recursion with coaxial and dangle stacking. The $R^{cd}_{\text{INTER}}(i, j, \phi)$ recursion references $Q^b_{i,d}$ elements that are computed using either the R^b_{INTRA} recursion of Figure S18 (if i and d are on the same strand) or the R^s_{INTER} recursion of Figure S27 (if i and d are on different strands). $R^b_{\text{INTER}}(i, j, \phi)$ operates on a conditional ensemble for subsequence $[i, j]$ with i and j on different strands and base paired to each other (depicted with a solid line between i and j). The function COMPLEMENTARY(ϕ_i, ϕ_j) checks if bases ϕ_i and ϕ_j are complementary (Watson–Crick or wobble pair) without regard to whether i and j are sufficiently separated along the strand to be able to pair sterically. The recursion distinguishes five cases that are combined using $\oplus$ in the recursion equation:

- *Exterior loop with coaxial stacking on terminal base pair $j \cdot i$:* the exterior loop with two or more terminal base pairs and coaxial stacking on terminal base pair $j \cdot i$.

  - *Base cases.* The base cases correspond to the recursion diagrams on the first row of Figure S27 and are treated using term C_1 in the subroutine MULTICOAXINTER (recursion equation S58). The diagram on the left treats the case where terminal base pair $j \cdot i$ forms a coaxial stack with adjacent terminal base pair $d+1 \cdot j-1$, depicted as a dotted straight line between i and $d+1$. The contribution of subsequence $[i+1, d]$ is incorporated by element $Q^n_{i+1,d}$. The pale pink shading corresponds to the sequence-dependent penalties for two terminal base pairs in an exterior loop context, $\Delta G^{\text{terminalbp}}_{d+1,j-1}(\phi)$ and $\Delta G^{\text{terminalbp}}_{j,i}(\phi)$ (note that the indices are ordered j then i to reflect $5'$ to $3'$ from the perspective of the exterior loop). The dark pink shading corresponds to the sequence-dependent coaxial stacking recursion energy $\Delta G^{\text{coax}}_{d+1,j-1,i}(\phi)$ (dependent on the sequences of base pairs $d+1 \cdot j-1$ and $j \cdot i$). Note that $\Delta G^{\text{coax}}_{d+1,j-1,i}(\phi)$ requires only 3 indices because j is implied by $j-1$. The recursion on the right treats the analogous



$$R^b_{\text{INTER}}(i,j,\phi) \equiv \begin{cases} C_1, & \textsc{Complementary}(\phi_i, \phi_j) \\ \mathbb{0}, & \text{otherwise} \end{cases}$$

$$\begin{aligned} C_1 \equiv\ & \textsc{ExteriorCoaxInter}(i,j,\phi,\eta) \oplus \textsc{ExteriorDangleInter}(i,j,\phi,\eta) \\ & \oplus \textsc{InteriorInter}(i,j,\phi,\eta) \\ & \oplus \textsc{MultiCoaxInter}(i,j,\phi,\eta) \oplus \textsc{MultiDangleInter}(i,j,\phi,\eta) \end{aligned}$$

Figure S27: R^b_{INTER} recursion with coaxial and dangle stacking. Top: recursion diagram. Bottom: recursion equation.

$$\textsc{ExteriorCoaxInter}(i,j,\phi,\eta) \equiv C_1 \otimes W(\Delta G^{\text{terminalbp}}_{j,i}(\phi)) \oplus \begin{cases} C_2 \otimes W(\Delta G^{\text{terminalbp}}_{j,i}(\phi)), & i+1 < m \text{ and } j-1 \geq n \\ (C_3 \oplus C_5) \otimes W(\Delta G^{\text{terminalbp}}_{j,i}(\phi)), & i+1 = m \text{ and } j-1 \geq n \\ (C_4 \oplus C_6) \otimes W(\Delta G^{\text{terminalbp}}_{j,i}(\phi)), & i+1 < m \text{ and } j = n \\ \mathbb{0}, & \text{otherwise} \end{cases}$$

where $m = \textsc{First}(\eta)$

$n = \textsc{Last}(\eta)$

$$C_1 \equiv \bigoplus_{\overline{d} \in \textsc{Valid}(i+1,j_{1,\eta})} \left[\textsc{dot}\left(Q^n_{i+1,\overline{d}}, Q^b_{\overline{d}+1,j-1}, W(\Delta G^{\text{coax}}_{\overline{d}+1,j-1,i}(\phi) + \Delta G^{\text{terminalbp}}_{\overline{d}+1,j-1}(\phi))\right) \oplus \textsc{dot}\left(Q^b_{i+1,\overline{d}}, Q^n_{\overline{d}+1,j-1}, W(\Delta G^{\text{coax}}_{j,i,\overline{d}}(\phi) + \Delta G^{\text{terminalbp}}_{i+1,\overline{d}}(\phi))\right)\right]$$

$$C_2 \equiv \bigoplus_{c \in \eta} \left[Q^{\varnothing}_{i+1,c-1} \otimes Q^b_{c,j-1} \otimes W(\Delta G^{\text{coax}}_{c,j-1,i}(\phi) + \Delta G^{\text{terminalbp}}_{c,j-1}(\phi)) \oplus Q^b_{i+1,c-1} \otimes Q^{\varnothing}_{c,j-1} \otimes W(\Delta G^{\text{coax}}_{j,i,c-1}(\phi) + \Delta G^{\text{terminalbp}}_{i+1,c-1}(\phi))\right]$$

$$C_3 \equiv \bigoplus_{\overline{d} \in \textsc{Valid}(i,j-1,\eta)} \textsc{dot}\left(Q^{\varnothing}_{i+1,\overline{d}}, Q^b_{\overline{d}+1,j-1}, W(\Delta G^{\text{coax}}_{\overline{d}+1,j-1,i}(\phi) + \Delta G^{\text{terminalbp}}_{\overline{d}+1,j-1}(\phi))\right)$$

$$C_4 \equiv \bigoplus_{\overline{d} \in \textsc{Valid}(i+1,j,\eta)} \textsc{dot}\left(Q^b_{i+1,\overline{d}}, Q^{\varnothing}_{\overline{d}+1,j-1}, W(\Delta G^{\text{coax}}_{j,i,\overline{d}}(\phi) + \Delta G^{\text{terminalbp}}_{i+1,\overline{d}}(\phi))\right)$$

$$C_5 \equiv Q^b_{i+1,j-1} \otimes W(\Delta G^{\text{coax}}_{i+1,j-1,i}(\phi) + \Delta G^{\text{terminalbp}}_{i+1,j-1}(\phi))$$

$$C_6 \equiv Q^b_{i+1,j-1} \otimes W(\Delta G^{\text{coax}}_{j,i,j-1}(\phi) + \Delta G^{\text{terminalbp}}_{i+1,j-1}(\phi)).$$

(S61)

$$\textsc{ExteriorDangleInter}(i,j,\phi,\eta) \equiv \bigoplus_{\substack{k\in\{0,1\}\\ l\in\{0,1\}}} \begin{cases} C_1, & i+k+1 < m \text{ and } j-l-1 \geq n \\ C_2, & i+k+1 = m \text{ and } j-l-1 \geq n \\ C_2, & i+k+1 < m \text{ and } j-l = n \\ C_3, & i+k+1 = m \text{ and } j-l = n \text{ and } m = n \\ \mathbb{0}, & \text{otherwise} \end{cases}$$

$$\begin{aligned} \text{where } m &= \textsc{First}(\eta) \\ n &= \textsc{Last}(\eta) \\ C_1 &\equiv Q^{n}_{i+k+1,j-l-1} \otimes C_3 \\ C_2 &\equiv Q^{\varnothing}_{i+k+1,j-l-1} \otimes C_3 \\ C_3 &\equiv W(\Delta G^{\text{dangle}}_{j-l,j,i,i+k}(\phi) + \Delta G^{\text{terminalbp}}_{j,i}(\phi)) \end{aligned} \tag{S62}$$

case where terminal base pair $j \cdot i$ forms a coaxial stack with adjacent terminal base pair $i+1 \cdot d$. To ensure that disconnected structures are excluded from the ensemble, the function Valid returns the set of valid vectorization ranges for which nucleotides d and $d+1$ are on the same strand (i.e., such that d and $d+1$ do not take on values that would place a nick between them).

- *Edge cases.* In the base case, the Q^{n} element incorporates a nick with a neighboring $Q^{\varnothing}$ element on either side. The edge cases diagrammed in rows 2 to 4 of Figure S27 correspond to states where either: one of the neighboring $Q^{\varnothing}$ elements is omitted because the nick is adjacent to one of the two coaxially-stacking base pairs (row 2 corresponding to term C_2 and row 3 corresponding to terms C_3 and C_4), or where both of the neighboring $Q^{\varnothing}$ elements are omitted because the nick is adjacent to both of the coaxially stacking base pairs (row 4 corresponding to terms C_5 and C_6). To ensure that disconnected structures are excluded from the ensemble for terms C_3 and C_4, the function Valid returns the set of valid vectorization ranges for which nucleotides d and $d+1$ are on the same strand (i.e., such that d and $d+1$ do not take on values that would place a nick between them).

- *Exterior loop with a dangle stacking state involving terminal base pair $j \cdot i$:* the exterior loop with one or more terminal base pairs and a dangle stacking state involving terminal base pair $i \cdot j$.

  - *Base case.* The base case corresponds to the leftmost recursion diagram in row 5 of Figure S27 and is treated by term C_1 in the subroutine MultiDangleInter

(recursion equation S58). The contribution of subsequence $[i+k+1, j-l-1]$ is incorporated by element $Q^n_{i+k+1,j-l-1}$. The pale pink shading corresponds to the sequence-dependent penalty for a terminal base pair in an exterior loop context, $\Delta G^{\text{terminalbp}}_{j,i}(\phi)$ (note that the indices are ordered j then i to reflect 5′ to 3′ from the perspective of the multiloop). The medium pink shading corresponds to the sequence-dependent dangle stacking energy $\Delta G^{\text{dangle}}_{j-l,j,i,i+k}(\phi)$. Note that $k, l \in \{0, 1\}$ determine whether unpaired nucleotides dangle stack on the adjacent base pair $j \cdot i$. The situation is analogous to that in R^{cd}_{INTER} (as detailed in Figure S14) with the only difference being that in R^{cd}_{INTER}, $i + k$ and $j - l$ index the paired bases and here $i + k$ and $j - l$ index the unpaired bases.

  - *Edge cases.* In the base case, the Q^n element incorporates a nick with a neighboring $Q^{\varnothing}$ element on either side. The edge cases in diagrams 2 to 4 of row 5 of Figure S27 correspond to states where either: one of the neighboring $Q^{\varnothing}$ elements is omitted because the nick is adjacent to the dangle stacking state on one side (diagrams 2 and 3 corresponding to term C_2), or where both of the neighboring $Q^{\varnothing}$ elements are omitted because the nick is adjacent to the dangle stacking state on both sides (diagram 4 corresponding to term C_3).

- *Interior loop:* the interior loop closed by the two terminal base pairs $i \cdot j$ and $d \cdot e$ (depicted by straight solid lines). Calculation of the interior loop contributions using an $O(N^4)$ or $O(N^3)$ version of the INTERIORINTER recursion is described in Section S2.4. This treatment of interior loops is the same as for the non-stacking recursions.

- *Multiloop with coaxial stacking on terminal base pair $j \cdot i$:* the multiloop closed by three or more terminal base pairs with coaxial stacking on base pair $j \cdot i$. This case corresponds to the two recursion diagrams on the seventh row of Figure S27 and is treated by the subroutine MULTICOAXINTER (recursion equation S63). The recursion on the left treats the case where terminal base pair $j \cdot i$ forms a coaxial stack with adjacent terminal base pair $d + 1 \cdot j - 1$, depicted as a dotted straight line between i and $d + 1$. The contribution of subsequence $[d + 1, j - 1]$ is incorporated by element $Q^b_{d+1,j-1}$. The contributions of one or more remaining stacking states in subsequence $[i + 1, d]$ are incorporated by element $Q^m_{i+1,d}$. The pale green shading corresponds to three multiloop recursion energies: 1) the penalty for formation of a multiloop $\Delta G^{\text{multi}}_{\text{init}}$, 2) the sequence-independent penalties for two terminal base pairs in a multiloop, $\Delta G^{\text{multi}}_{\text{bp}}$ (corresponding to base pairs $d + 1 \cdot j - 1$ and $j \cdot i$), 3) the sequence-dependent penalties for two terminal base pairs in a multiloop context,

$\Delta G^{\text{terminalbp}}_{d+1,j-1}(\phi)$ and $\Delta G^{\text{terminalbp}}_{j,i}(\phi)$ (note that the indices are ordered j then i to reflect 5′ to 3′ from the perspective of the multiloop). The dark green shading corresponds to the sequence-dependent coaxial stacking recursion energy $\Delta G^{\text{coax}}_{d+1,j-1,i}(\phi)$ (dependent on the sequences of base pairs $d+1 \cdot j-1$ and $j \cdot i$). Note that $\Delta G^{\text{coax}}_{d+1,j-1,i}(\phi)$ requires only 3 indices because j is implied by $j-1$. To exclude exterior loop states that are not treated by this multiloop recursion, the function VALID returns the set of valid vectorization ranges for which nucleotides d and $d+1$ are on the same strand (i.e., such that d and $d+1$ do not take on values that would place a nick between them). The recursion on the right treats the analogous case where terminal base pair $j \cdot i$ forms a coaxial stack with adjacent terminal base pair $i+1 \cdot d$.

- *Multiloop with dangle stacking on terminal base pair $j \cdot i$:* the multiloop closed by three or more terminal base pairs with a dangle stacking state involving base pair $j \cdot i$. This case corresponds to the two recursion diagrams on the eighth row of Figure S18 and is treated by the subroutine MULTIDANGLEINTER (recursion equation S64).

  - *Base case with two or more additional stacking states.* The recursion on the left treats the case where there is a dangle stacking state involving the terminal base pair $j \cdot i$ (depicted as a dotted straight line between $i+k$ and $j-l$) and a 3′-most stacking state in subsequence $[d+1, j-l-1]$ (depicted as a dashed line between $d+1$ and $j-l-1$). The contributions for subsequence $[d+1, j-l-1]$ are incorporated using a $Q^{ms}_{d+1,j-l-1}$ element. The pale green shading corresponds to four multiloop recursion energies: 1) the penalty for formation of a multiloop $\Delta G^{\text{multi}}_{\text{init}}$, 2) the sequence-independent penalty for one terminal base pair in a multiloop, $\Delta G^{\text{multi}}_{\text{bp}}$ (corresponding to base pair $j \cdot i$), 3) the penalty per unpaired nucleotide in a multiloop $\Delta G^{\text{multi}}_{\text{nt}}$ (a total of $k+l$ dangling nucleotides; as a result this term is zeroed out when $k=l=0$). 4) the sequence-dependent penalty for a terminal base pair in a multiloop context, $\Delta G^{\text{terminalbp}}_{j,i}(\phi)$ (note that the indices are ordered j then i to reflect 5′ to 3′ from the perspective of the multiloop). The medium green shading corresponds to the sequence-dependent dangle stacking recursion energy $\Delta G^{\text{dangle}}_{j-l,j,i,i+k}(\phi)$. To exclude exterior loop states that are not treated by this multiloop recursion, the function VALID returns the set of valid vectorization ranges for which nucleotides d and $d+1$ are on the same strand (i.e., such that d and $d+1$ do not take on values that would place a nick between them). Note that $k, l \in \{0, 1\}$ determine whether unpaired nucleotides dangle stack on the adjacent base pair $j \cdot i$ in a multiloop context. The situation is analogous to that in R^{cd}_{INTER} (as detailed in Figure S14) with the only difference

being that in R^{cd}_{INTER}, $i + k$ and $j - l$ index the paired bases and here $i + k$ and $j - l$ index the unpaired bases.

- *Edge case with one additional coaxial stacking state.* The recursion on the right treats the case where there is a dangle stacking state involving the terminal base pair $j \cdot i$ (depicted as a dotted straight line between $i + k$ and $j - l$) and one coaxial stacking state in subsequence $[e, j - l - 1]$ (depicted as a dashed line between e and $j - l - 1$). The contributions for subsequence $[e, j - l - 1]$ are incorporated using a $Q^{mcs}_{e,j-l-1}$ element. The pale green shading corresponds to four multiloop recursion energies: 1) the penalty for formation of a multiloop $\Delta G^{\text{multi}}_{\text{init}}$, 2) the sequence-independent penalty for one terminal base pair in a multiloop, $\Delta G^{\text{multi}}_{\text{bp}}$ (corresponding to base pair $j \cdot i$), 3) the penalty per unpaired nucleotide in a multiloop $\Delta G^{\text{multi}}_{\text{nt}}$ ($k + l$ dangling nucleotides plus the unpaired nucleotides $i+k+1, \ldots, e-1$; as a result this term is zeroed out when $k = l = 0$ and $e = i + 1$). 4) the sequence-dependent penalty for a terminal base pair in a multiloop context, $\Delta G^{\text{terminalbp}}_{j,i}(\phi)$ (note that the indices are ordered j then i to reflect 5′ to 3′ from the perspective of the multiloop). The medium green shading corresponds to the sequence-dependent dangle stacking recursion energy $\Delta G^{\text{dangle}}_{j-l,j,i,i+k}(\phi)$. Note that this edge case covers the scenario where there are exactly three terminal base pairs and the two pairs that are not $j \cdot i$ are coaxially stacked. That situation is not covered by the base case because for that recursion, a multiloop with 3 terminal base pairs would have one terminal base pair in the Q^{ms} element, and one terminal base pair in the Q^{m} element (hence, those two terminal base pairs cannot coaxially stack since they are in different recursions).

Note that unlike the R^{b}_{INTRA} recursion of Figure S18, for R^{b}_{INTER} there is no hairpin loop case as i and j are on different strands.

$$\textsc{MultiCoaxInter}(i,j,\phi,\eta) \equiv C_1 \otimes W(\Delta G^{\text{multi}}_{\text{init}} + 2\Delta G^{\text{multi}}_{\text{bp}} + \Delta G^{\text{terminalbp}}_{j,i}(\phi))$$

where $m = \textsc{First}(\eta)$

$n = \textsc{Last}(\eta)$

$$C_1 \equiv \bigoplus_{\overline{d} \in \textsc{Valid}(i+1,j-1,\eta)} \Bigg[\textsc{dot}\left(Q^{b}_{i+1,\overline{d}}, Q^{m}_{\overline{d}+1,j-1}, W(\Delta G^{\text{coax}}_{j,i,\overline{d}}(\phi) + \Delta G^{\text{terminalbp}}_{i+1,\overline{d}}(\phi))\right) \oplus \textsc{dot}\left(Q^{m}_{i+1,\overline{d}}, Q^{b}_{\overline{d}+1,j-1}, W(\Delta G^{\text{coax}}_{\overline{d}+1,j-1,i}(\phi) + \Delta G^{\text{terminalbp}}_{\overline{d}+1,j-1}(\phi))\right) \Bigg] \tag{S63}$$

$$\textsc{MultiDangleInter}(i,j,\phi,\eta) \equiv \bigoplus_{\substack{k\in\{0,1\}\\ l\in\{0,1\}}} \begin{cases} C_1 \oplus C_2, & i+k+1 < m \text{ and } j-l-1 \geq n \\ \mathbb{0}, & \text{otherwise} \end{cases}$$

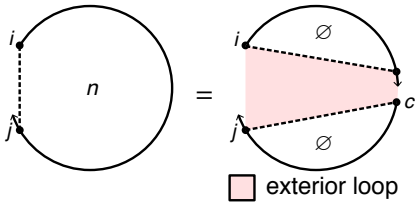
where $m = \textsc{First}(\eta)$

$n = \textsc{Last}(\eta)$

$$C_1 \equiv \bigoplus_{\overline{d} \in \textsc{Valid}(i+k+1,j-l-1,\eta)} \Bigg[\textsc{dot}\left(Q^{m}_{i+k+1,\overline{d}}, Q^{ms}_{\overline{d}+1,j-l-1}\right) \otimes W(\Delta G^{\text{dangle}}_{j-l,j,i,i+k}(\phi) + \Delta G^{\text{multi}}_{\text{init}} + \Delta G^{\text{multi}}_{\text{bp}} + (k+l)\Delta G^{\text{multi}}_{\text{nt}} + \Delta G^{\text{terminalbp}}_{j,i}(\phi)) \Bigg]$$

$$C_2 \equiv \textsc{dot}\left(Q^{mcs}_{\overline{e},j-l-1}\right) \otimes W(\Delta G^{\text{dangle}}_{j-l,j,i,i+k}(\phi) + \Delta G^{\text{multi}}_{\text{init}} + \Delta G^{\text{multi}}_{\text{bp}} + \overline{n}_{\text{nt}}\Delta G^{\text{multi}}_{\text{nt}} + \Delta G^{\text{terminalbp}}_{j,i}(\phi))$$

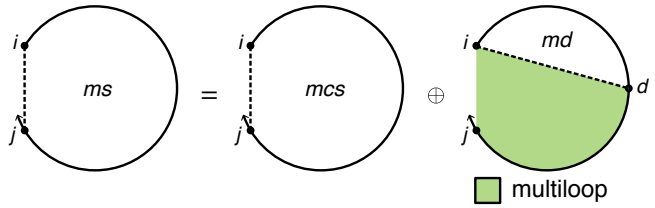
$$\overline{e} \equiv [i+k+1 : m-1], \quad \overline{n}_{\text{nt}} \equiv [k+l : m-i+l-2]. \tag{S64}$$



$$R^{n}_{\text{INTER}}(i,j,\phi) \equiv \bigoplus_{c\in\eta} Q^{\varnothing}_{i,c-1} \otimes Q^{\varnothing}_{c,j}$$

Figure S28: R^{n}_{INTER} recursion with coaxial and dangle stacking. Top: recursion diagram. Bottom: recursion equation.

R^{n}_{INTER} recursion with coaxial and dangle stacking. The $R^{b}_{\text{INTER}}(i,j,\phi)$ recursion references Q^{n} elements that are computed using the R^{n}_{INTER} recursion displayed in Figure S28. $R^{n}_{\text{INTER}}(i,j,\phi)$ operates on a conditional ensemble in an exterior loop context with a nick at c. For each nick $c \in \eta$, the contribution of subsequence $[i, c-1]$ is incorporated by element $Q^{\varnothing}_{i,c-1}$ and the contribution of subsequence $[c, j]$ is incorporated by element $Q^{\varnothing}_{c,j}$. Shading denotes no recursion energies. Note that while the subsequence $[i, j]$ is disonnected within the $R^{n}_{\text{INTER}}(i,j,\phi)$ recursion due to the nick at c, these states are connected when used in the context of the R^{b}_{INTER} recursion.



$$R^{ms}_{\text{INTER}}(i,j,\phi) \equiv Q^{mcs}_{i,j} \oplus \text{DOT}\left(Q^{md}_{i,\overline{d}}, W(\overline{n}_{\text{nt}}\Delta G^{\text{multi}}_{\text{nt}})\right)$$
$$\text{where } \overline{d} \equiv [\text{LAST}(\eta) : j], \quad \overline{n}_{\text{nt}} \equiv [0 : j - \text{LAST}(\eta)]^{r}$$

Figure S29: R^{ms}_{INTER} recursion with coaxial and dangle stacking. Top: recursion diagram. Bottom: recursion equation.

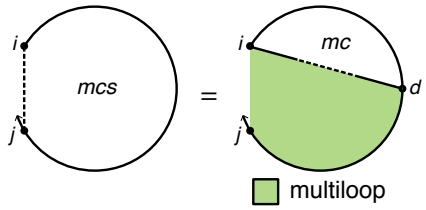
R^{ms}_{INTER} recursion with coaxial and dangle stacking. The $R^{b}_{\text{INTER}}(i,j,\phi)$ recursion references $Q^{ms}_{d+1,j-l-1}$ elements that are computed using either the R^{ms}_{INTRA} recursion shown of Figure S19 (if $d+1$ and $j-l-1$ are on the same strand) or the R^{ms}_{INTER} recursion of Figure S29

(if $d+1$ and $j-l-1$ are on different strands). $R^{ms}_{\text{INTER}}(i,j,\phi)$ operates on a conditional ensemble for subsequence $[i,j]$ in a multiloop context containing one stacking state starting at i and ending in the interval $[i+1,j]$ (depicted as a dashed line between i and j). There are two cases that are combined using $\oplus$ in the recursion equation:

- *Coaxial stacking state:* The contribution for the coaxial stacking state in subsequence $[i,j]$ is calculated using a $Q^{mcs}_{i,j}$ element.
- *Dangle stacking state:* The contribution for the dangle stacking state in subsequence $[i,d]$ is incorporated using a $Q^{md}_{i,d}$ element. Shading corresponds to the recursion energy penalty per unpaired nucleotide in a multiloop, $\Delta G^{\text{multi}}_{\text{nt}}$ (nucleotides $d+1,\ldots,j$ for a total of $j-d$ unpaired nucleotides; as a result, this term is zeroed out in the edge case where $d=j$). Note that in the dot product the range multiplying $\Delta G^{\text{multi}}_{\text{nt}}$ runs in reverse order because the number of unpaired nucleotides, $j-d$, decreases in size as d increases in size. Nucleotide d must always be on the last strand ($d \geq \text{LAST}(\eta)$) to ensure that there are no nicks in the subsequence $[d,j]$, which would lead to either a disconnected structure (which is not permitted in the complex ensemble) or an exterior loop state (which is not handled by this recursion).

Note that R^{ms} directly incorporates the R^{mcs} recursion which serves as an efficiency wrapper of the R^{mc} recursion, and hence, R^{ms} is an efficiency wrapper of R^{mc} (the 3′-most coaxial stacking state in a multiloop context). Note also that R^{ms} is an efficiency wrapper of the R^{md} recursion (the 3′-most dangle stacking state in a multiloop context). Taken together R^{mc} and R^{md} represent the 3′-most stacking state in a multiloop context, analogous to R^{cd} representing the 3′-most stacking state (coaxial or dangle) in an exterior loop context. The reason that R^{mc} (coaxial stacking states) and R^{md} (dangle stacking states) are calculated and stored separately in a multiloop context, is that coaxial-only information (stored in element Q^{mcs}) is needed for the edge case previously described for the right recursion diagram in the bottom row of Figure S27. As a result, coaxial-only information is calculated using the efficiency wrapper R^{mcs} for use in that edge case, and then coaxial-only and dangle-only information are combined by the R^{ms} efficiency wrapper (which is fully analogous to the R^{s} efficiency wrapper in the exterior loop context). With this approach, the operations spent calculating coaxial stacking information for Q^{mcs} elements are not repeated when calculating both coaxial and dangle stacking for Q^{ms} elements.

R^{mcs}_{INTER} recursion with coaxial and dangle stacking. The $R^{ms}_{\text{INTER}}(i,j,\phi)$ recursion references Q^{mcs} elements that are computed using the $R^{mcs}_{\text{INTER}}(i,j,\phi)$ recursion displayed in



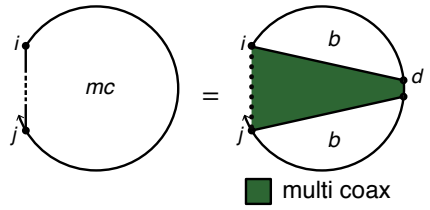
$$R^{mcs}_{\text{INTER}}(i,j,\phi) \equiv \text{DOT}\left(Q^{mc}_{i,\overline{d}}, W(\Delta G^{\text{multi}}_{\text{nt}}\overline{n}_{\text{nt}}\right)$$

$$\text{where } \overline{d} \equiv [\text{LAST}(\eta) : j], \quad \overline{n}_{\text{nt}} \equiv [0 : j - \text{LAST}(\eta)]^r$$

Figure S30: R^{mcs}_{INTER} recursion with coaxial and dangle stacking. Top: recursion diagram. Bottom: recursion equation.

Figure S30. $R^{mcs}_{\text{INTER}}(i,j,\phi)$ operates on a conditional ensemble for subsequence $[i,j]$ in a multiloop context containing one coaxial stacking state starting at i and ending in the interval $[i+1,j]$ (depicted as a dashed line between i and j). The contribution for the coaxial stacking state in subsequence $[i,d]$ is incorporated using a $Q^{mc}_{i,d}$ element. Shading denotes the penalty per unpaired nucleotide in a multiloop $\Delta G^{\text{multi}}_{\text{nt}}$ (the unpaired nucleotides $d+1,\ldots,j$; as a result this term is zeroed out when $d=j$). Note that in the dot product the range multiplying $\Delta G^{\text{multi}}_{\text{nt}}$ runs in reverse order because the number of unpaired nucleotides, $j-d$, decreases in size as d increases in size. Nucleotide d must always be on the last strand to ensure that there are no nicks in the subsequence $[d,j]$, which would lead to either a disconnected structure (which is not permitted in the complex ensemble) or an exterior loop state (which is not handled by this multiloop recursion). Note that the R^{mcs} recursion serves as an efficiency wrapper of the R^{mc} recursion (here, representing the 3′-most coaxial stacking state in a multiloop context).

R^{mc}_{INTER} recursion with coaxial and dangle stacking. The $R^{mcs}_{\text{INTER}}(i,j,\phi)$ recursion references Q^{mc} elements that are computed using the $R^{mc}_{\text{INTER}}(i,j,\phi)$ recursion displayed in Figure S31. This recursion treats a single coaxial stacking state in a multiloop context (depicted as a straight line between i and j that is solid at both ends and dashed in the middle to indicate that i and j are both base-paired but not to each other). Two adjacent terminal base pairs ($i \cdot d$ and $d+1 \cdot j$) coaxially stack on each other. The contributions of subsequences $[i,d]$ and $[d+1,j]$ are incorporated using $Q^{b}_{i,d}$ and $Q^{b}_{d+1,j}$ elements. Shading corresponds to three kinds of recursion energy: 1) the sequence-independent penalties for two terminal base pairs in a multiloop, $\Delta G^{\text{multi}}_{\text{bp}}$ (corresponding to base pairs $i \cdot d$ and



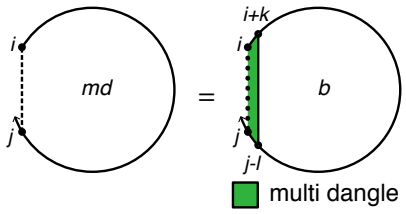
$$R^{mc}_{\text{INTER}}(i,j,\phi) \equiv \bigoplus_{\overline{d}\in \text{VALID}(i,j,\eta)} \text{DOT}\left(Q^{b}_{i,\overline{d}}, Q^{\overline{b}}_{\overline{d}+1,j}, C_1\right) \otimes W(2\Delta G^{\text{multi}}_{\text{bp}})$$

$$\text{where } C_1 \equiv W(\Delta G^{\text{coax}}_{i,\overline{d},j}(\phi) + \Delta G^{\text{terminalbp}}_{i,\overline{d}}(\phi) + \Delta G^{\text{terminalbp}}_{\overline{d}+1,j}(\phi))$$

Figure S31: R^{mc}_{INTER} recursion with coaxial and dangle stacking. Top: recursion diagram. Bottom: recursion equation.

$d + 1 \cdot j$), 2) the sequence-dependent penalties for two terminal base pairs in a multiloop context, $\Delta G^{\text{terminalbp}}_{i,d}(\phi)$ and $\Delta G^{\text{terminalbp}}_{d+1,j}(\phi)$ (dependent on the sequence of base pairs $i \cdot d$ and $d+1 \cdot j$), 3) the sequence-dependent coaxial stacking free energy $\Delta G^{\text{coax}}_{i,d,j}(\phi)$ (dependent on the sequences of base pairs $i \cdot d$ and $d+1 \cdot j$). Note that $\Delta G^{\text{coax}}_{i,d,j}(\phi)$ requires only 3 indices because $d + 1$ is implied by d. The function VALID returns the set of valid vectorization ranges for which nucleotides d and $d + 1$ are on the same strand (i.e., such that d and $d + 1$ do not take on values that would place a nick between them) to avoid an exterior loop state (which is not handled by this multiloop recursion).

R^{md}_{INTER} recursion with coaxial and dangle stacking. The $R^{ms}_{\text{INTER}}(i,j,\phi)$ recursion references Q^{md} elements that are computed using the R^{md}_{INTER} recursion displayed in Figure S32. $R^{md}_{\text{INTER}}(i,j,\phi)$ treats a single dangle stacking state (depicted as a dotted line between i and j) in a multiloop context with either zero, one, or two unpaired nucleotides dangle stacking on an adjacent terminal base pair ($i + k \cdot j - l$). The recursion diagram represents these four alternative dangle stacking states corresponding to no dangles, 5′ dangle, 3′ dangle, or terminal mismatch (see Figure S14 for details). The contribution of subsequence $[i+k, j-l]$ is incorporated using a $Q^{b}_{i+k,j-l}$ element. Shading corresponds to four recursion energies: 1) the sequence-independent penalty for one terminal base pair in a multiloop, $\Delta G^{\text{multi}}_{\text{bp}}$ (corresponding to base pair $i + k \cdot j - l$), 2) the penalty per unpaired nucleotide in a multiloop $\Delta G^{\text{multi}}_{\text{nt}}$ (a total of $k + l$ dangling nucleotides; as a result this term is zeroed out when $k = l = 0$). 3) the sequence-dependent penalty for a terminal base pair in a multiloop loop context, $\Delta G^{\text{terminalbp}}_{i+k,j-l}(\phi)$ (dependent on the sequence of base pair $i + k \cdot j - l$), 4) the



$$R^{md}_{\text{INTER}}(i,j,\phi) \equiv \bigoplus_{\substack{k\in\{0,1\}\\ l\in\{0,1\}}} \begin{cases} C_1, & i+k<m \text{ and } j-l \geq n \\ \mathbb{0}, & \text{otherwise} \end{cases}$$

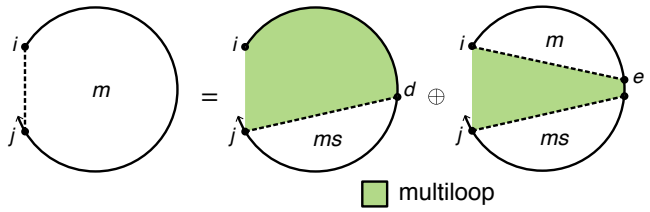
where $m = \text{FIRST}(\eta)$

$n = \text{LAST}(\eta)$

$$C_1 \equiv Q^b_{i+k,j-l} \otimes W(\Delta G^{\text{dangle}}_{i,i+k,j-l,j}(\phi) + \Delta G^{\text{multi}}_{\text{bp}} + (k+l)\Delta G^{\text{multi}}_{\text{nt}} + \Delta G^{\text{terminalbp}}_{i+k,j-l}(\phi))$$

Figure S32: R^{md}_{INTER} recursion with coaxial and dangle stacking. Top: recursion diagram. Bottom: recursion equation.

sequence-dependent dangle stacking free energy $\Delta G^{\text{dangle}}_{i,i+k,j-l,j}(\phi)$ which takes on one of four values corresponding to the four dangle stacking states (see Figure S14). The index limits in the recursion equation prevent a nick between a dangling nucleotide and the base pair on which it stacks (i.e., no nick between i and $i+k$, and no nick between j and $j-l$) to avoid an exterior loop state (which is not handled by this multiloop recursion).



$$\begin{aligned} R^{m}_{\text{INTER}}(i,j,\phi) \equiv & \bigoplus_{\overline{d}\in \text{VALID}(i,j,\eta)} \text{DOT}\left(Q^{m}_{i,\overline{d}}, Q^{ms}_{\overline{d}+1,j}\right) \\ & \oplus \text{DOT}\left(Q^{ms}_{\overline{e},j}, W(\overline{n}_{\text{nt}}\Delta G^{\text{multi}}_{\text{nt}})\right) \end{aligned} \tag{S65}$$

$$\text{where } \overline{e} \equiv [i : \text{FIRST}(\eta)], \quad \overline{n}_{\text{nt}} \equiv [0 : \text{FIRST}(\eta) - i - 1]$$

Figure S33: R^{m}_{INTER} recursion with coaxial and dangle stacking. Top: recursion diagram. Bottom: recursion equation.

R^m_{INTER} recursion with coaxial and dangle stacking. The $R^b_{\text{INTER}}(i, j, \phi)$ recursion references Q^m elements that are computed using either the R^m_{INTRA} recursion of Figure S23 (if the end points of the referenced subsequence are on the same strand), or the R^m_{INTER} recursion of Figure S33 (if the end points of the referenced subsequence are on different strands). $R^m_{\text{INTER}}(i, j, \phi)$ operates on a conditional ensemble for subsequence $[i, j]$ in a multiloop context where i and j may or may not be paired (depicted with a dashed line between i and j in the recursion diagram) and where there is at least one stacking state in the subsequence. This recursion distinguishes two cases that are combined using $\oplus$ in the recursion equation:

- *One stacking state:* the case where there is exactly one stacking state in subsequence $[i, j]$ in a multiloop context. This stacking state starts at d and ends in the interval $[d + 1, j]$ (depicted by a straight dashed line between d and j); the contribution of subsequence $[d, j]$ is incorporated by element $Q^{ms}_{d,j}$. Shading corresponds to the recursion energy, $\Delta G^{\text{multi}}_{\text{nt}}$, representing the penalty per unpaired nucleotide in a multiloop (nucleotides $i, \ldots, d-1$ for a total of $d-i$ unpaired nucleotides; as a result, this term is zeroed out in the edge case where $d = i$). Nucleotide d must always be on the first strand to ensure that there are no nicks in the subsequence $[i, d]$, which would lead to either a disconnected structure (which is not permitted in the complex ensemble) or an exterior loop state (which is not handled by this multiloop recursion).

- *More than one stacking state:* the case where there are two or more stacking states in subsequence $[i, j]$ in a multiloop context. The 3′-most stacking state starts at $e+1$ and ends in the interval $[e+2, j]$ (depicted by a straight dashed line between $e+1$ and j); the contribution of subsequence $[e + 1, j]$ is incorporated by element $Q^{ms}_{e+1,j}$. There are one or more additional stacking states in the interval $[i, e]$ (the straight dashed line denotes that i and e may or may not be paired); the contribution of subsequence $[i, e]$ is incorporated by element $Q^m_{i,e}$. The shading does not represent any recursion energies as all multiloop contributions are handled by other recursions: 1) there are no terminal base pairs in a multiloop context explicitly defined in this case, 2) there are no unpaired bases in a multiloop context explicitly defined in this case. To exclude exterior loop states that are not treated by this multiloop recursion, the function VALID returns the set of valid vectorization ranges for which nucleotides e and $e + 1$ are on the same strand (i.e., such that e and $e + 1$ do not take on values that would place a nick between them).

S3 Evaluation algebras for each physical quantity

To calculate each physical quantity, the generic recursions of Section S2 are combined with a quantity-specific *evaluation algebra* (summarized in Table 2.2). Here, we provide additional details on the evaluation algebra abstraction, and on the definition of evaluation algebras for different physical quantities.

An evaluation algebra defines the mathematical form of the generic operators that appear in recursion equations. We define an evaluation algebra $\mathcal{A}$ to have the following properties:

1. $D_{\mathcal{A}}$ is the domain of values in the evaluation algebra.
2. $\oplus_{\mathcal{A}}$ is an operation yielding a combination of alternative conditional ensembles. Thus, $c = a \oplus_{\mathcal{A}} b$ reflects the notion of "c contains either conditional ensemble a or conditional ensemble b."
3. $\otimes_{\mathcal{A}}$ is an operation that yields a composition of conditional ensembles. Thus, $c = a \otimes_{\mathcal{A}} b$ reflects the notion of "c contains both conditional ensemble a and conditional ensemble b."
4. $\mathbb{0}_{\mathcal{A}}$ is a value in $D_{\mathcal{A}}$ that satisfies the concept of additive identity. Physically, $\mathbb{0}_{\mathcal{A}}$ represents an impossible substructure (i.e., a structural element that is not in the complex ensemble).
5. $\mathbb{1}_{\mathcal{A}}$ is a value in $D_{\mathcal{A}}$ that satisfies the concept of multiplicative identity. Physically, $\mathbb{1}_{\mathcal{A}}$ represents a structure in the free energy reference state.
6. $W_{\mathcal{A}}$ is an operation that takes a free energy to a value in $D_{\mathcal{A}}$. Physically, $W_{\mathcal{A}}$ represents the weight on an individual substructure.
7. $Q_{\mathcal{A}}$ is an operation that yields a value in $D_{\mathcal{A}}$ from an recursion element in the set of all recursion elements Λ. $Q_{\mathcal{A}}$ is used to give the prior-calculated result over a given conditional ensemble.

As such, an evaluation algebra may be classified as an algebraic semiring equipped with two additional unary operators W and Q. We typically elide the dependence on $\mathcal{A}$ below to simplify the notation. We now describe the definitions of these properties for evaluation algebras corresponding to different physical quantities, treating the calculation of scalar quantities in Section S3.1 and the calculation of quantities requiring structure generation in Section S3.2.

S3.1 Evaluation algebras for scalar outputs

S3.1.1 SumProduct: sum product evaluation algebra

Within evaluation algebra $\mathcal{A}$ = SumProduct, the partition function of an ensemble, $\overline{Q}(\phi)$, is computed by taking the sum over products of Boltzmann factors.

$$\begin{aligned} D &= \mathbb{R}_{\geq 0} \\ a \oplus b &= a + b \\ a \otimes b &= a \cdot b \\ \mathbb{0} &= 0 \\ \mathbb{1} &= 1 \\ W(g) &= \exp\left(\frac{-g}{k_B T}\right) \\ Q(\lambda) &= A_{i,j} \text{ where } \lambda = (A, i, j) \text{ and } A \text{ is the stored recursion matrix.} \end{aligned} \tag{S66}$$

Each expression in the algebra represents a Boltzmann factor, which is necessarily a non-negative real number. An impossible structure maps to a Boltzmann factor of 0, whereas a structure with a zero reference free energy maps to a Boltzmann factor of 1.

S3.1.2 Count: structure count evaluation algebra

Within evaluation algebra $\mathcal{A}$ = Count, the size of an ensemble, $\overline{\Gamma}$ or $\overline{\Gamma}^{|}$, is computed by taking the sum over products of subensemble sizes.

$$\begin{aligned} D &= \mathbb{Z}_{\geq 0} \\ a \oplus b &= a + b \\ a \otimes b &= a \cdot b \\ \mathbb{0} &= 0 \\ \mathbb{1} &= 1 \\ W(g) &= 1 \\ Q(\lambda) &= A_{i,j} \text{ where } \lambda = (A, i, j) \text{ and } A \text{ is the stored recursion matrix.} \end{aligned} \tag{S67}$$

The only difference between Count and SumProduct is the definition of W. While the domain of Count is theoretically non-negative integers, it is still implemented using floating point types to avoid integer overflow.

S3.1.3 MinSum: minimum sum evaluation algebra

Within evaluation algebra $\mathcal{A}$ = MinSum, the free energy of the minimum free energy (MFE) stacking state, $\overline{\Delta G}(\phi, s^{||}_{\text{MFE}})$, is the minimum over sums of conditional ensemble free energies.

$$\begin{aligned} D &= \mathbb{R} \cup \{\infty\} \\ a \oplus b &= \min(a, b) \\ a \otimes b &= a + b \\ \mathbb{0} &= \infty \\ \mathbb{1} &= 0 \\ W(g) &= g \\ Q(\lambda) &= A_{i,j} \text{ where } \lambda = (A, i, j) \text{ and } A \text{ is the stored recursion matrix.} \end{aligned} \tag{S68}$$

An impossible structure is assigned a free energy of ∞.

S3.1.4 SplitExp: overflow-safe evaluation algebra

Here, we expand our description in Table 2.2 of the overflow-safe evaluation algebra $\mathcal{A}$ = SplitExp. For exposition, that description was a simplification of the production evaluation algebra, which must be implemented somewhat differently as we now discuss. The main text description includes a free parameter γ representing the negative exponent of the output variable. Each Boltzmann factor is then evaluated relative to γ. Here, to factor out γ, we lift our evaluation algebra into a set of higher order functions. Thus, instead of each expression being a pair of numbers, each expression is itself a function returning its associated mantissa value and its offset exponent relative to the input γ. We use the anonymous form of function notation $x \mapsto y$ to notate a function taking x and returning y.

$$
\begin{aligned}
D &= \mathbb{Z} \mapsto \mathbb{R}_{\geq 0} \times \mathbb{Z} \\
a \oplus b &= \gamma \mapsto \left(a_{\mathrm{m}}(\gamma) \cdot 2^{a_{\mathrm{e}}(\gamma)} + b_{\mathrm{m}}(\gamma) \cdot 2^{b_{\mathrm{e}}(\gamma)}, 0\right) \\
a \otimes b &= \gamma \mapsto \left(a_{\mathrm{m}}(b_{\mathrm{e}}(\gamma)) \cdot b_{\mathrm{m}}(\gamma), a_{\mathrm{e}}(b_{\mathrm{e}}(\gamma))\right) \\
\mathbb{0} &= \gamma \mapsto (0, 0) \\
\mathbb{1} &= \gamma \mapsto (1, \gamma) \\
W(g) &= \gamma \mapsto \left(\exp\left(\frac{-g}{k_B T}\right), \gamma\right) \\
Q(\lambda) &= \gamma \mapsto \left(M_{i,j}, E_{i,j} + \gamma\right) \text{ where } \lambda = (A, i, j) \text{ and } M, E \text{ are the} \\
&\qquad \text{stored recursion matrices for } A \text{ of mantissas and exponents, respectively.}
\end{aligned}
\tag{S69}
$$

An element a returns, as a function of γ, the mantissa and exponent values expressed respectively as $a_{\mathrm{m}}(\gamma)$ and $a_{\mathrm{e}}(\gamma)$. An element a may be converted to the domain of SumProduct using the transformation $a_{\mathrm{m}}(\gamma)2^{a_{\mathrm{e}}(\gamma)-\gamma}$. With infinite-precision arithmetic, we can plug in any value $\gamma \equiv \gamma_0$ to perform a calculation. Using finite-precision arithmetic, however, γ must be chosen judiciously to avoid floating point overflow. We describe our method of choosing γ_0 below.

Addition works by aligning both expressions to the output exponent γ and then adding the resultant mantissas. As the output mantissa has been aligned to γ, the output shift exponent is 0. For example, take $a = \mathbb{1}$, $b = W(g_0)$. Then

$$
\begin{aligned}
a \oplus b &= (a_{\mathrm{m}}(\gamma) \cdot 2^{a_{\mathrm{e}}(\gamma)} + b_{\mathrm{m}}(\gamma) \cdot 2^{b_{\mathrm{e}}(\gamma)}, 0), \\
&= \left(1 \cdot 2^{\gamma} + \exp\left(\frac{-g_0}{k_B T}\right) \cdot 2^{\gamma}, 0\right), \\
&= \left(2^{\gamma}\left(1 + \exp\left(\frac{-g_0}{k_B T}\right)\right), 0\right).
\end{aligned}
\tag{S70}
$$

Multiplication works by multiplying the mantissas and adding the exponents. The exponent shift is applied to only one quantity; therefore, the shift is applied directly to b, the result of which is propagated to a. (This choice of ordering could be inverted without changing the output result.) For example, take $a = Q^b_{p,q}$, $b = Q^m_{r,s}$. Then

$$
\begin{aligned}
a \otimes b &= (a_{\mathrm{m}}(\gamma) \cdot b_{\mathrm{m}}(\gamma) \cdot M^m_{r,s}, a_{\mathrm{e}}(E^m_{r,s} + \gamma)), \\
&= (M^b_{p,q} \cdot M^m_{r,s}, E^b_{p,q} + E^m_{r,s} + \gamma).
\end{aligned}
\tag{S71}
$$

In our implementation, mantissa and exponent values of the same bit width are held in separate arrays M and E for each recursion matrix. Single-precision floating point and signed integers are used, such that the total storage cost of this method is identical to running SumProduct in double precision. From the output expression of a given recursion R in SplitExp, the following output numbers are calculated:

$$\begin{aligned} m_\lambda &= R_{\mathrm{m}}(\lambda, \phi)(\gamma_0(\lambda)) \\ e_\lambda &= R_{\mathrm{e}}(\lambda, \phi)(\gamma_0(\lambda)) - \gamma_0(\lambda, \phi). \end{aligned} \tag{S72}$$

Now, using the function frexp canonically defined such that $\mathrm{frexp}_{\mathrm{m}}(x)2^{\mathrm{frexp}_{\mathrm{e}}(x)} = x$ and either $\frac{1}{2} \leq \mathrm{frexp}_{\mathrm{m}}(x) < 1$ or $\mathrm{frexp}_{\mathrm{m}}(x) = \mathrm{frexp}_{\mathrm{e}}(x) = 0$, the respective entry in the recursion matrix is set via the following convention:

$$\begin{aligned} M(\lambda) &\leftarrow \mathrm{frexp}_{\mathrm{m}}(m_\lambda) \\ E(\lambda) &\leftarrow \mathrm{frexp}_{\mathrm{e}}(m_\lambda) + e_\lambda. \end{aligned} \tag{S73}$$

To prevent overflow from occurring, if an expression has a theoretical partition function of q, γ_0 should be relatively close to $-\log_2 q$. Specifically, $|\gamma_0 + \log_2 q|$ should be less than the maximum floating point exponent in the given arithmetic (128 for 32-bit precision, 1024 for 64-bit precision). For recursion element $\lambda = (A, i, j)$, we choose γ_0 as follows from the matrix E containing the exponents of A:

$$\begin{aligned} \gamma_0(\lambda) &\equiv \min_{\substack{i \leq d \leq e \leq j \\ (d,e) \neq (i,j)}} -E_{d,e} \\ &= \begin{cases} \min\left(-E_{i+1,j}, -E_{i,j-1}\right) & i < j \\ 0 & \text{otherwise.} \end{cases} \end{aligned} \tag{S74}$$

In other words, γ_0 is based on the exponents of the two known adjacent elements in the matrix. Although it possible to try multiple choices of γ_0 as a failsafe, in practice, the single definition of γ_0 above is sufficient to avoid overflow from occurring for all test cases in our validate suite (Section S7).

S3.1.5 LogSum: log semiring evaluation algebra as alternative overflow-safe approach

For completeness, we outline the possibility of using the log semiring LogSum to avoid overflow in partition function computation. In this evaluation algebra, each quantity a corresponds to quantity 2^a in SumProduct. (The base-2 logarithm is used for computational convenience.)

$$\begin{aligned} D_{\text{LogSum}} &= \mathbb{R} \cup \{-\infty\} \\ \oplus_{\text{LogSum}}(a,b) &= \log_2(2^a + 2^b) = \max(a,b) + \log_2(2^{a-\max(a,b)} + 2^{b-\max(a,b)}) \\ \otimes_{\text{LogSum}}(a,b) &= a + b \\ \mathbb{0}_{\text{LogSum}}(a,b) &= -\infty \\ \mathbb{1}_{\text{LogSum}} &= 0 \\ W_{\text{LogSum}}(g) &= -(kT \log 2)^{-1} g \\ Q(\lambda) &= A_{i,j} \text{ where } \lambda = (A,i,j) \text{ and } A \text{ is the stored recursion matrix.} \end{aligned} \tag{S75}$$

In practice, this evaluation algebra proved to be simpler but less efficient than SplitExp. Within a given dot product of many contributions, first the maximum contribution must be computed beforehand across all contributions, then the adjusted exponentiations of each contribution must be calculated, and finally the exponentiations must be summed. Even after optimization and vectorization, we found that LogSum was >2× more expensive than SplitExp in partition function computations due to the need for floating point exponentiation and two separate scans through the arrays of contributions.

S3.2 Evaluation algebras for structure generation

Structure generation conceptually yields specific secondary structures from a given weighting on the ensemble $\overline{\Gamma}$ or $\overline{\Gamma}^{||}$. In this case, because any given structure depends on only a sparse subset of recursion matrix elements, a backtracking operation order is in general more efficient than a forward pass iteration. Such an operation order jumps between recursion elements in an opposite direction to that in the forward pass. To enable such an approach, the recursion matrices in the forward pass must be computed beforehand (e.g., calculating the MFE before generating an ensemble of suboptimal structures or calculating the partition function before Boltzmann sampling structures).

Here, we correspondingly distinguish between a *forward* evaluation algebra and a *backtracking* evaluation algebra. Whereas a forward evaluation algebra like SumProduct operates

on a subset of $\mathbb{R}$, we define a backtracking evaluation algebra to operate on a domain of conditional ensembles which may be queried for a set of dependent recursion elements. This ordering may be viewed as equivalent to the topological ordering of the directed acyclic graph of computed quantities in a forward dynamic program (e.g., in Figure 2.8). In all cases, any conditional ensemble s containing a recursion element (b, i, j) indicates that $i \cdot j$ is a base pair in s, a feature which is used to output final structures from the algorithm. We next illustrate how a backtracking evaluation algebra may be defined with respect to the associated forward evaluation algebra.

S3.2.1 Generic approach to structure generation

We start with a consideration of the simplest structure generation evaluation algebras. To simplify the exposition, in Table 2.2 we defined evaluation algebras ArgRand to calculate a single randomly sampled structure and ArgMin to determine the MFE stacking state, $s^{||}_{\text{MFE}}$, assuming it was unique. For a given element a, each evaluation algebra was defined such that a was a pair of scalar value a_{v} and recursion element set a_λ.

In Table 2.2, the operations on a_{v} and b_{v} in ArgRand duplicate the operations of SumProduct, whereas the operations on a_{v} and b_{v} in ArgMin duplicate the operations of MinSum. For ArgRand and ArgMin, the operations on a_λ and b_λ are the same for $\otimes$, which represents the set union $\cup$ of recursion elements that occur in the same conditional ensemble. However, the operations on a_λ and b_λ are different in the definition of $\oplus$, which is responsible for attributing the scalar contribution $a_{\text{v}} \oplus b_{\text{v}}$ to an individual conditional ensemble a_λ or b_λ. In ArgRand, $\oplus$ yields a random weighted choice via

$$\arg\,\text{rand}(a, b) \equiv \left(a_\lambda \text{ if } \mathscr{U}(0, a_{\text{v}} + b_{\text{v}}) < a_{\text{v}} \text{ else } b_\lambda\right) \tag{S76}$$

where $\mathscr{U}$ is the random uniform distribution function. In contrast, in ArgMin, $\oplus$ yields the conditional ensemble which is lower in free energy via

$$\arg\,\min(a, b) \equiv (a_\lambda \text{ if } a_{\text{v}} < b_{\text{v}} \text{ else } b_\lambda)\,. \tag{S77}$$

Thus we can see that an intuitive approach for constructing a backtracking evaluation algebra is to augment a corresponding forward evaluation algebra with customized operations for structure attribution. We next describe the resulting definitions of ArgRand and ArgMin.

S3.2.2 ArgRand: single Boltzmann sample evaluation algebra

Within evaluation algebra $\mathcal{A} = \textsc{ArgRand}$, each element a is a pair of partition function value a_{v} and set of recursion elements a_λ.

$$\begin{aligned} D &= \mathbb{R}_{\geq 0} \times \mathscr{P}(\Lambda) \\ a \oplus b &= \left(a_{\mathrm{v}} + b_{\mathrm{v}}, \arg\,\mathrm{rand}(a, b)\right) \\ a \otimes b &= (a_{\mathrm{v}} \cdot b_{\mathrm{v}}, a_\lambda \cup b_\lambda) \\ \mathbb{0} &= (0, \varnothing) \\ \mathbb{1} &= (1, \varnothing) \\ W(g) &= \left(\exp\left(\frac{-g}{k_B T}\right), \varnothing\right) \\ Q(\lambda) &= \left(Q_{\textsc{SumProduct}}(\lambda), \{\lambda\}\right). \end{aligned} \tag{S78}$$

Operations on the first element, a_{v}, are defined using SumProduct. The second element, a_λ, is a set of recursion elements defining a restricted ensemble of conditional ensemble compositions. The set of all possible sets of recursion elements λ is denoted as $\mathscr{P}(\Lambda)$. Any quantity that does not depend on the output of another recursion is thus assigned $a_\lambda = \varnothing$.

S3.2.3 ArgMin: unique MFE structure evaluation algebra

Within evaluation algebra $\mathcal{A} = \textsc{ArgMin}$, each element a is a pair of value a_{v} and set of recursion elements a_λ.

$$\begin{aligned} D &= \mathbb{R} \cup \{\infty\} \times \mathscr{P}(\Lambda) \\ a \oplus b &= \left(\min(a_{\mathrm{v}}, b_{\mathrm{v}}), \arg\,\min(a, b)\right) \\ a \otimes b &= (a_{\mathrm{v}} + b_{\mathrm{v}}, a_\lambda \cup b_\lambda) \\ \mathbb{0} &= (\infty, \varnothing) \\ \mathbb{1} &= (0, \varnothing) \\ W(g) &= \left(g, \varnothing\right) \\ Q(\lambda) &= \left(Q_{\textsc{MinSum}}(\lambda), \{\lambda\}\right). \end{aligned} \tag{S79}$$

Operations on the first element, a_{v}, are defined using MinSum. The second element, a_λ, is a set of recursion elements defining a restricted ensemble of conditional ensemble

compositions. The set of all possible sets of recursion elements λ is denoted as $\mathscr{P}(\Lambda)$. Any quantity that does not depend on the output of another recursion is thus assigned $a_\lambda = \varnothing$.

S3.2.4 Efficient structure generation via lazy evaluation

We derived more programmatically efficient evaluation algebras for generating and ensemble of J Boltzmann-sampled structures, $\overline{\Gamma}_{\text{sample}}(\phi, J)$, or for generating a ensemble of suboptimal structures, $\overline{\Gamma}_{\text{subopt}}(\phi, \Delta G_{\text{gap}})$. Note that MFE proxy structure(s) can be generated by setting $\Delta G_{\text{gap}} = 0$. Here, we describe efficient evaluation algebras for ArgRandJ and ArgMinGap that build upon ArgRand and ArgMin.

The simpler but less efficient algebras ArgRand and ArgMin yield full representations of the chosen conditional ensembles, which are then enqueued by the respective operation order algorithms. Our more efficient algorithms work by backtracking through a given recursion element and enqueueing any combinations of recursion elements in conditional ensembles that match a given criterion. The matching evaluation algebras incorporate the enqueueing operation κ directly such that each piece of a conditional ensemble is enqueued immediately as it is encountered. These evaluation algebras are similarly generic but operate lazily on recursion elements (obviating storage of intermediate structures which might not affect the final results).

We describe the ArgRandJ and ArgMinGap evaluations using a generic framework defined with respect to a given forward algebra (SumProduct and MinSum, respectively). As in Section S3.1.4, we make use of the anonymous form of function notation $x \mapsto y$ to notate a function taking x and returning y. Formally, we define the enqueueing operation κ recursively as a function in $D_\kappa \equiv (\mathbb{R}, \mathscr{P}(\Lambda)) \mapsto D_\kappa$; effectively, it may be viewed as an iterator across each alternative conditional ensemble. Let $\mathcal{B}$ be the backtracking evaluation algebra and $\mathcal{A}$ the forward algebra. Then we classify each expression in $\mathcal{B}$ as a closure within $D_\mathcal{B} \equiv D_\kappa \mapsto D_\kappa$ and denote a set of recursion elements as $\Lambda_i \in \mathscr{P}(\Lambda)$ below.

Within the evaluation algebra, addition of a and b intuitively represents the successive iteration through multiple alternative structures a and b. It may be defined formally as a higher-order function that accomplishes functional composition:

$$\oplus_\mathcal{B}(a, b) = \kappa \mapsto b(a(\kappa)). \tag{S80}$$

Multiplication of a and b represents the independent combinations of conditional ensembles from a and b being composed together–in effect, a lazily evaluated outer product of

conditional ensembles within a and b. It may be defined formally as the higher-order function:

$$\otimes_{\mathcal{B}}(a, b) = \kappa \mapsto a\big((x_1, \Lambda_1) \mapsto b\big((x_2, \Lambda_2) \mapsto \kappa\big(x_1 \otimes_{\mathcal{A}} x_2, \Lambda_1 \cup \Lambda_2\big)\big)\big). \tag{S81}$$

The properties of commutativity and associativity are preserved for $\oplus_{\mathcal{B}}$ and $\otimes_{\mathcal{B}}$ so long as κ is independent of the order of evaluation (i.e., $\kappa(x_1, \Lambda_1)(x_2, \Lambda_2) = \kappa(x_2, \Lambda_2)(x_1, \Lambda_1)$), a property that is satisfied by all algorithms discussed here. The multiplicative identity corresponds to a zero free energy structure, which is not dependent on any recursion elements:

$$\mathbb{1}_{\mathcal{B}} = \kappa \mapsto \kappa(\mathbb{1}_{\mathcal{A}}, \varnothing). \tag{S82}$$

The additive identity is defined as the identity function, reflecting an impossible structure by returning the enqueueing function unchanged:

$$\mathbb{0}_{\mathcal{B}} = \kappa \mapsto \kappa. \tag{S83}$$

$W_{\mathcal{B}}$ brings a free energy parameter into the evaluation algebra domain, and is not dependent on any recursion elements:

$$W_{\mathcal{B}}(g) = \kappa \mapsto \kappa\left(W_{\mathcal{A}}(g), \varnothing\right). \tag{S84}$$

Finally, the recursion matrix operator yields the forward algebra value and a singleton of its associated recursion element:

$$Q_{\mathcal{B}}(\lambda) = \kappa \mapsto \kappa\left(Q_{\mathcal{A}}(\lambda), \{\lambda\}\right). \tag{S85}$$

See Sections S4.4 and S4.6 for specifications of the enqueing function κ used in Boltzmann sampling and suboptimal structure generation, respectively.

S3.2.5 ArgRandJ: simultaneous Boltzmann sample evaluation algebra

The implemented evaluation algebra $\mathcal{A}$ = ArgRandJ that uses higher order functions to accomplish lazy evaluation is a specialization of the generic structure generation algebra (Section S3.2.4) for the associated forward evaluation algebra SumProduct. See Section S4.4 for the definition of κ.

$$\begin{aligned}
D &= D_\kappa \mapsto D_\kappa \\
a \oplus b &= \kappa \mapsto b(a(\kappa)) \\
a \otimes b &= \kappa \mapsto a\big((x_1, \Lambda_1) \mapsto b\big((x_2, \Lambda_2) \mapsto \kappa\big(x_1 \cdot x_2, \Lambda_1 \cup \Lambda_2\big)\big)\big) \\
\mathbb{0} &= \kappa \mapsto \kappa \\
\mathbb{1} &= \kappa \mapsto \kappa(1, \varnothing) \\
W(g) &= \kappa \mapsto \kappa\left(\exp\left(\frac{-g}{k_B T}\right), \varnothing\right) \\
Q(\lambda) &= \kappa \mapsto \kappa(Q_{\text{SumProduct}}(\lambda), \{\lambda\}).
\end{aligned} \tag{S86}$$

To avoid overflow issues for large complexes, we extended the above evaluation algebra in a similar fashion to that described in Section S3.1.4.

S3.2.6 ArgMinGap: suboptimal structure evaluation algebra

The implemented evaluation algebra $\mathcal{A}$ = ArgMinGap that uses higher order functions to accomplish lazy evaluation is a specialization of the generic structure generation algebra (Section S3.2.4) for the associated forward evaluation algebra MinSum. See Section S4.6 for the definition of κ.

$$\begin{aligned}
D &= D_\kappa \mapsto D_\kappa \\
a \oplus b &= \kappa \mapsto b(a(\kappa)) \\
a \otimes b &= \kappa \mapsto a\big((x_1, \Lambda_1) \mapsto b\big((x_2, \Lambda_2) \mapsto \kappa\big(x_1 + x_2, \Lambda_1 \cup \Lambda_2\big)\big)\big) \\
\mathbb{0} &= \kappa \mapsto \kappa \\
\mathbb{1} &= \kappa \mapsto \kappa(0, \varnothing) \\
W(g) &= \kappa \mapsto \kappa(g, \varnothing) \\
Q(\lambda) &= \kappa \mapsto \kappa(Q_{\text{MinSum}}(\lambda), \{\lambda\}).
\end{aligned} \tag{S87}$$

S4 Operation orders for each physical quantity

S4.1 A partial order on recursion elements

Here, we describe novel operation orders that take advantage of the blockwise approach to calculations in the multistranded ensemble (Figure S34). The resultant dependency graph of recursion elements provides the main constraints in correctly handling calculations for a given physical quantity. Let λ denote a recursion element such that it holds all of the non-global information needed to address a recursion (e.g., λ could be represented as $(m, 2, 5)$ for element $Q^m_{2,5}$). We define a partial order on any two recursion elements λ_1, λ_2 such that if (and only) if the definition of recursion λ_2 is dependent on that for λ_1, then $\lambda_1 < \lambda_2$. We define this partial order as a lexicographical order on (1) the strand indices of the recursions, (2) the subsequence indices of the recursions, and (3) the recursion types. In other words, two recursions are to be compared based on their associated strand indices, then their subsequence indices, then their recursion types.

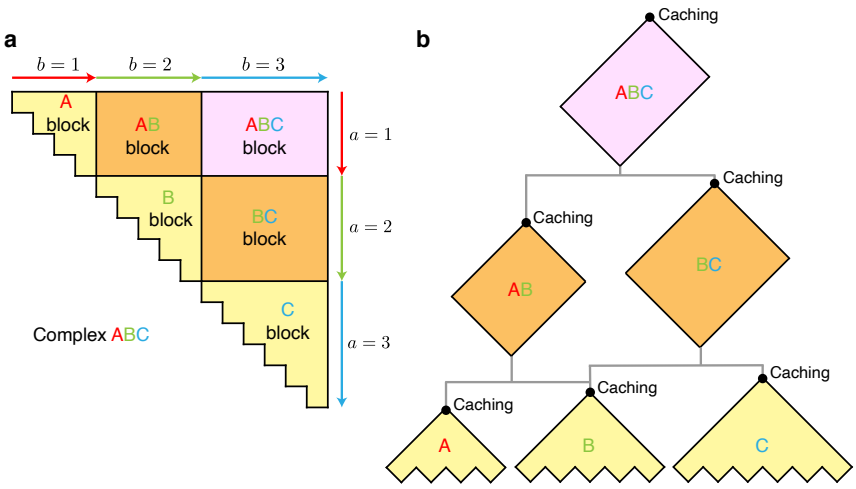


Figure S34: Blockwise operation order. (a) Depiction of the blockwise approach. Strand indices a and b are used in the pseudocode of Sections S4.2 and S4.3. (b) Dependency graph for block evaluation: bottom to top for forward algorithms, top to bottom for backtracking algorithms. Black circles denote locations in the forward algorithms where block results may be cached to avoid recomputation. Analogous to a single stranded dynamic program, which uses subproblems on subsequences of nucleotides, the multistranded dynamic program uses subproblems on subsequences of strands.

Ordering on strand indices. We developed dynamic programming algorithms working over subsequences of strands within a multistranded complex. Let a_x and b_x be the sorted

strand indices of a given recursion element λ_x. Then we implement a partial order on two recursion elements λ_1 and λ_2 by defining that $\lambda_1 < \lambda_2$ if ($a_2 < a_1$ and $b_1 \leq b_2$) or ($a_2 = a_1$ and $b_1 < b_2$). For instance, if $a_1 = 2$, $b_1 = 3$, $a_2 = 1$, and $b_2 = 3$, then $\lambda_1 < \lambda_2$ because the strand range $[a_1 : b_1]$ is nested within $[a_2 : b_2]$.

Ordering on subsequence indices. Next we incorporate an analogous partial order on the subsequence indices of different recursion elements. We define the subsequence index of a nucleotide as the index of its position within its strand. Let i_x and j_x be the sorted subsequence indices of a given recursion λ_x. Then, if the strand indices of λ_1, λ_2 are equal, we define that $\lambda_1 < \lambda_2$ if ($i_2 < i_1$ and $j_1 \leq j_2$) or ($i_2 = i_1$ and $j_1 < j_2$). For instance, if (1) the strand indices of λ_1 and λ_2 are equal, and (2) if $i_1 = 10$, $j_1 = 30$, $i_2 = 5$, and $j_2 = 40$, then $\lambda_1 < \lambda_2$ because the nucleotide range $[i_1 : j_1]$ is nested within $[i_2 : j_2]$. This ordering mirrors the structure encountered with strand indices and is the historical order implicit in dynamic programming algorithms working within a single-stranded ensemble.

Ordering on recursion types. Finally, we define an ordering on the recursion types of different recursion elements. Let i and j be any fixed sequence indices. Then we define $\mathbf{T}$ to be an ordered sequence of recursion types such that for any indices $p < q$, the output of recursion $(\mathbf{T}_p, i, j)$ is not dependent on that of $(\mathbf{T}_q, i, j)$. Next, if T_1 and T_2 are two recursion types, then we define that $T_1 < T_2$ if and only if T_1 appears before T_2 in $\mathbf{T}$. There are multiple logically consistent sequences $\mathbf{T}$ which could be defined for a given set of recursions. We implemented the following ones for the recursions of Section S2:

$$\begin{aligned} \mathbf{T}_{\text{nostacking}} &\equiv [x, b, ms, m, s, \varnothing] \\ \mathbf{T}_{\text{stacking}} &\equiv [x, b, md, mc, mcs, ms, cd, s, m, \varnothing, n] \end{aligned} \tag{S88}$$

For example, consider the two recursion elements $\lambda_1 \equiv (b, 1, 5)$ (corresponding to $Q^b_{1,5}$) and $\lambda_2 \equiv (m, 1, 5)$ (corresponding to $Q^m_{1,5}$). Then $\lambda_1 < \lambda_2$ because b appears before m in $\mathbf{T}$, no matter which set of recursions is used.

S4.2 Operation order for partition function, structure count, and MFE

Here, we describe the operation order for a block-based dynamic program over subcomplexes used for partition function, structure count, and MFE. It relies on separate subroutines to calculate triangular intrastrand blocks and rectangular interstrand blocks.

ComplexDynamicProgram takes parameters $\mathcal{A}$, the evaluation algebra, ϕ, the sequence of the complex for which to compute the partition function, and C, a map from sequences to blocks in which to store and retrieve computed blocks. $\mathcal{A}$ may be one of SumProduct, Count, MinSum, or SplitExp. The subroutine returns the complete block of dynamic program results FullQ.

Algorithm S3: Blockwise operation order over subcomplexes.

ComplexDynamicProgram($\mathcal{A}, \phi, C$)

```
1  L = Number of Sequences(φ)
2  FullQ ← EmptyBlock(Length(φ)) ▷ Initialize all matrix storage
3
4  for l ∈ [0 : L]
5      for a ∈ [1 : L − l]
6          b ← a + l
7          if φ^{a,b} ∈ C
8              ▷ Take result for block from cache
9              FullQ_{a,b} ← C[φ^{a,b}]
10         else
11             if a = b
12                 ▷ Calculate intrastrand block
13                 FullQ_{a,a} ← IntraStrandDynamicProgram(A, φ^{a,a})
14             else
15                 ▷ Calculate interstrand block
16                 InterStrandDynamicProgram(A, φ^{a,b}, FullQ_{[a:b],[a:b]})
17             ▷ Put result for block into cache
18             C[φ^{a,b}] ← FullQ_{a,b}
19 return FullQ
```

The outermost element of FullQ corresponding to $Q_{1,N}$ may be post-processed into the target physical quantity as described in Section S5.

Operation order for intrastrand blocks. We define the subsidiary operation order as follows for a single-stranded subcomplex to return a fresh block Q. No prior information from other blocks is necessary. Iteration proceeds in a forward sweep as illustrated in Figure 2.4.

Algorithm S4: Operation order for a triangular intrastrand block.

IntraStrandDynamicProgram($\mathcal{A}, \phi$)

```
1  N ← Length(φ)
2  for l ∈ [1 : N]
3      for i ∈ [1 : N − l]
4          j ← i + l − 1
5          for T ∈ T
6              ▷ Calculate and store recursion output for Q^T_{i,j}
7              λ ← (T, i, j)
8              Q(λ) ← R_Intra(λ, φ)
9  return Q
```

Operation order for interstrand blocks. We define the subsidiary operation order for a multistranded subcomplex to update the parameter Q with the outermost block, given that all subsidiary blocks are already calculated. For instance, in Figure S34a, InterStrand-DynamicProgram would calculate the top-right block ABC using the prior calculations of blocks A, B, C, AB, and BC.

Algorithm S5: Operation order for a rectangular interstrand block.

InterStrandDynamicProgram($\mathcal{A}, \phi, Q$)

$N \leftarrow$ Length(ϕ)
$b \leftarrow$ Nicks(ϕ)
$m \leftarrow$ First(b) ▷ index of first base on second strand
$n \leftarrow$ Last(b) ▷ index of first base on last strand

▷ Iteration proceeds from lowest to highest $l \equiv j - i + 1$
for $l \in [n - m + 1 : N]$
    for $i \in [\max(l, n) - l + 1 : \min(m, N - l)]$
        $j \leftarrow i + l - 1$
        for $T \in \mathbf{T}$
            ▷ Calculate and store recursion output for $Q^T_{i,j}$
            $\lambda \leftarrow (T, i, j)$
            $Q(\lambda) \leftarrow R_{\text{Inter}}(\lambda, \phi)$

S4.3 Operation order for equilibrium pair probability matrix

Here, we present the operation order for a block-based backtrack-free equilibrium pair probability algorithm (see Figure 2.14b). Symbols have the same meanings as in Section S4.2, except FullQ is the block for the doubled sequence ϕ' instead of the input ϕ. Similarly, the Q and Q^b matrices in line 27 refer to the recursion matrices for ϕ'. After the dynamic programming algorithm the resultant Q and Q^b entries are post-processed into the pair probabilities matrix as in equation (2.17). The output of the algorithm is the matrix $\overline{P}(\phi)$, such the $\overline{P}_{i,j}(\phi)$ is the equilibrium probability of base pair $i \cdot j$ in the distinguishable ensemble $\overline{\Gamma}$. It is interesting to note that the same operation order coupled with the MinSum evaluation algebra can be used to calculate a matrix $\overline{G}(\phi)$ such that $\overline{G}_{i,j}(\phi)$ is the lowest free energy of a stacking state containing base pair $i \cdot j$.

The subroutine PartialInterStrandDynamicProgram behaves identically to InterStrandDynamicProgram but stops once l in its outer recursion reaches $N \equiv \text{Length}(\phi)$. This saving can take place because the backtrack-free pair probabilities methodology (Figure 2.14) only requires results from element indices (i, j) such that $i \leq j \leq i + N$.

Algorithm S6: Operation order for backtrack-free pair probabilities.

PairProbabilities($\mathcal{A}, \phi,$ FullQ, C)

L = Number of Sequences(ϕ)
$\phi' = \phi + \phi$ ▷ Concatenated sequence of ϕ with itself
FullQ $\leftarrow$ EmptyBlock(Length(ϕ')) ▷ Initialize all matrix storage

for $l \in [0 : L]$
    for $a \in [1 : 2L - l]$
        $b \leftarrow a + l$
        if $\phi'^{a,b} \in C$
            ▷ Take result from cache
            FullQ$_{a,b} \leftarrow C[\phi'^{a,b}]$
        else
            if $a = b$
                ▷ Calculate intrastrand block
                FullQ$_{a,a}$ $\leftarrow$ IntraStrandDynamicProgram($\mathcal{A}, \phi'^{a}$)
            elseif $l < L$
                ▷ Calculate interstrand block
                InterStrandDynamicProgram($\mathcal{A}, \phi'^{a,b}$, FullQ$_{[a:b],[a:b]}$)
            else
                ▷ Calculate lower triangle of interstrand block
                PartialInterStrandDynamicProgram($\mathcal{A}, \phi'^{a,b}$, FullQ$_{[a:b],[a:b]}, N$)
            ▷ Put results in cache
            $C[\phi'^{a,b}] \leftarrow$ FullQ$_{a,b}$

Initialize $N \times N$ matrix P
for $i \in [1 : N]$
    for $j \in [1 : N]$ **and** $j \neq i$
        $P_{i,j} = Q^{b}_{i,j} Q^{b}_{j,N+i} Q^{-1}_{1,N}$ ▷ same as Equation 2.17
    $P_{i,i} = 1 - \sum_{1 \leq j \leq N, j \neq i} P_{i,j}$
return P

S4.4 Operation order for sampled structure generation

Here, we describe a new operation order for simultaneously Boltzmann sampling J structures with a worst-case time complexity $O(JN^2)$ using the full interior loop model. Algorithm illustrated in Figure S35 eliminates any recomputation of the same recursion element. A priority queue is defined via the partial order on recursion element λ from Section S4.1 via the recursion type, strand indices, and sequence indices of λ. The queue is initialized with the single recursion element $Q_{1,N}$ and the indices of the associated sampled structures $1, \ldots, J$ (i.e., all J sampled structures that are to be generated). When an element is popped from the priority queue, if J_λ of the sampled structures include this element, J_λ random numbers are drawn and sorted. Next, each of N_λ conditional ensembles is traversed exactly once, and each matching contribution is enqueued along with every index of a matched structure. This procedure yields a subproblem complexity of $O(N_\lambda + J_\lambda \log J_\lambda)$ compared to $O(J_\lambda N_\lambda)$ for the same algorithm run J times for a single sample (i.e., a sequential approach).

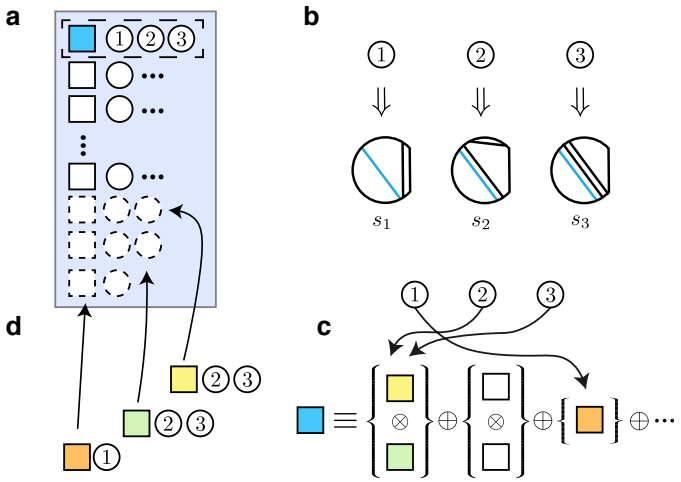


Figure S35: Illustration of simultaneous sampling operation order. (a) The top recursion element λ is popped off the priority queue along with its associated structures 1,2,3. (b) If the popped element λ is of type $Q^b_{d,e}$, a base pair between d and e is added to each associated structure 1,2,3. (c) The evaluation algebra is invoked with $J_\lambda = 3$, randomly assigning conditional ensembles to structures 1,2,3. (d) The recursion elements corresponding to each conditional ensemble are added to the priority queue along with their associated structures.

Algorithm S7 details the operation order for simultaneously generating J secondary structures Boltzmann sampled from the ensemble of a complex of N nucleotides with sequence

ϕ. The ArgRandJ evaluation algebra is used to backtrack through each contribution to a given recursion element. We achieve an $O(N_\lambda + J_\lambda \log J_\lambda)$ in the subproblem of backtracking through a given recursion element λ (neglecting logarithmic factors; see the complexity annotations in Step 4). The worst-case time complexity of the simultaneous sampling algorithm is $O(JN^2)$. Actual performance depends on the sequence of the complex. The speedup from simultaneous sampling is expected to be greatest when the Boltzmann ensemble is dominated by fewer conditional ensembles (e.g., when there is a deep well in the free energy landscape, as for designed ensembles), so that a simultaneous sampling approach avoids repeatedly sampling the same recursive elements, as would happen with a sequential sampling approach. Empirical measurements of the complete algorithm complexity are given in Section S6.6.3.

Algorithm S7: Operation order for simultaneous structure sampling.

1. Initialize an array L of J secondary structures with no base pairs.
2. Initialize an empty priority queue $\mathcal{P}$ of pairs of recursion element λ and vector of ordered structure indices $\vec{v}$.
3. Enqueue a pair of $\lambda = (\varnothing, 1, N)$ and $\vec{v} = [1 : J]$ into $\mathcal{P}$.
4. While $\mathcal{P}$ is not empty:
    a) $\boldsymbol{O(1)}$ **cost:** Dequeue the highest priority element λ and its respective indices $\vec{v}$ from $\mathcal{P}$ (Figure S35a).
    b) Let J_λ be the length of $\vec{v}$.
    c) $\boldsymbol{O(J_\lambda)}$ **cost:** If λ denotes any element $Q^b_{i,j}$, add a base pair $i \cdot j$ in each structure s_l for $l \in \vec{v}$ (Figure S35b).
    d) $\boldsymbol{O(J_\lambda)}$ **cost:** Initialize an array $\vec{w}$ of J_λ random numbers uniformly distributed between 0 and $Q_{\text{SumProduct}}(\lambda)$. $Q_{\text{SumProduct}}(\lambda)$ is the matrix element value obtained from the forward pass partition function calculation.
    e) $\boldsymbol{O(J_\lambda \log J_\lambda)}$ **cost:** Sort $\vec{w}$ and reorder $\vec{v}$ by the same permutation.
    f) Initialize $q = 0$ as the running sum of contributions and $k = 1$ as the running index.
    g) Calculate the generator $G = R_{\text{ArgRandJ}}(\lambda)(\kappa)$ in order to attribute conditional ensemble contributions to the output structure indices $\vec{v}$ (Figure S35c). Essentially, the generator achieves iteration over each possible conditional ensemble contribution to λ. For exposition, this may be achieved with the coroutine $\kappa(x, \Lambda)$ which yields (x, Λ) and returns κ. In practice, the loop below was programmatically implemented via a callback function.
    h) For each of N_λ contributions (x, Λ) in G until $k > J_\lambda$:
        i. Increment the accumulator $q \leftarrow q + x$.
        ii. $\boldsymbol{O(J_\lambda + N_\lambda)}$ **cost over all contributions:** Find the remaining weights below q by calculating $k' \leftarrow \text{UpperBound}(\vec{w}[k : j], q)$. This may be done via binary search.
        iii. $\boldsymbol{O(J_\lambda \log \min(J, N^2))}$ **cost over all contributions:** For each element λ in Λ, enqueue $(\lambda, \vec{v}[k : k' - 1])$ into $\mathcal{P}$ (Figure S35d). If λ was already present in $\mathcal{P}$, concatenate the indices $\vec{v}$ of the two items.
        iv. Update the running index $k \leftarrow k'$.
5. Return L.

S4.5 Operation order for interior loops in backtracking algorithms

We sample from the structural ensemble containing all interior loop states while achieving an asymptotic upper bound of $O(N^2)$ for a single sample. The same argument may be applied to determination of a unique MFE proxy structure, $s_{\text{MFE}'}$ (see Section S4.6). As we explain below, this complexity reduction is made possible by iterating through the interior loop states in order from fewest to most unpaired nucleotides. The operation orders for intrastrand recursions:

$$\textsc{InteriorBacktrackIntra}(i,j,\phi) \equiv \begin{cases} \bigoplus_{z=10}^{j-i-4} \bigoplus_{s=5}^{z-5} Q^b_{d,e} \otimes W(\Delta G^{\text{interior}}_{i,d,e,j}(\phi)), & j-i \geq 6 \\ \mathbb{0}, & \text{otherwise} \end{cases}$$
$$\text{where } d = i + z - s$$
$$e = j - s \tag{S89}$$

and interstrand recursions:

$$\textsc{InteriorBacktrackInter}(i,j,\phi) \equiv \begin{cases} \bigoplus_{z=10}^{j-n+m-i} \bigoplus_{r=\max(5,z-j+n)}^{\min(z-5,m-i-1)} \\ \qquad Q^b_{d,e} \otimes W(\Delta G^{\text{interior}}_{i,d,e,j}(\phi)), & i < m-1, n < j \\ \mathbb{0}, & \text{otherwise} \end{cases}$$
$$\text{where } d = i + r$$
$$e = j + r - z$$
$$m = \textsc{First}(\eta)$$
$$n = \textsc{Last}(\eta) \tag{S90}$$

contrast with the historical operation orders for interior loops, which consider all 5′ inner bases d in ascending order and then all 3′ inner bases e compatible with each d, again in ascending order (see equations (S47) and (S48)).

Note that like (S47) and (S48), the operation orders (S89) and (S90) result in $O(N^4)$ forward-pass algorithms (see Section S2.4). This follows because for each closing pair $i \cdot j$, we consider all $O(n^2)$ possible closing pairs $d \cdot e$, where $n = j - i + 1$. However, the new operation order nonetheless enables $O(N^2)$ single-sample performance, as we now show.

In the recursions for sampling the contributions to an element $Q^b_{i,j}$, hairpin loops, exterior loops, multiloops, and inextensible interior loops (including all bulge loops and stack loops) are all sampled first. From the R^b recursions in Figures S5, S10, S18, S27, one can see there are either $O(1)$ or $O(n)$ of these contribution types for a subsequence of length n. There is only one hairpin loop, one stack loop, and $O(n)$ bulge and inextensible interior

loops. While there are potentially more than $O(n)$ multiloops consistent with $i \cdot j$, they are handled recursively and there are only $O(n)$ contributions coming through Q^m elements. Therefore, if only these states are sampled, the algorithm will only recurse into at most $O(N)$ Q^b elements each costing at most $O(N)$ for an over all complexity of $O(N^2)$ and we would already have our bound.

So we limit ourselves to cases where at least one extensible interior loop is sampled. If iteration proceeds through these interior loops in ascending order of number of unpaired bases, each inner base pair $d \cdot e$ will be encountered at most once. To see this, assume the extensible interior loop with base pair $d \cdot e$ is sampled. Then every previous base pair $d' \cdot e'$ iterated through in order to reach $d \cdot e$ will meet one of the following conditions: $e' - d' > e - d$ or $d' < d < e' < e$. The first case occurs for all interior loops with fewer unpaired nucleotides than the loop bounded by base pair $d \cdot e$. The second case occurs for all interior loops with the same number of unpaired nucleotides as the loop bounded by base pair $d \cdot e$. In both cases, $\phi_{d',e'}$ is not a subsequence of $\phi_{d,e}$ and the base pair $d' \cdot e'$ cannot appear in $\phi_{d,e}$. Therefore, because (1) extensible interior loop contributions are only considered after contributions that lead to an overall asymptotic upper bound of $O(N^2)$, (2) base pairs bounding extensible loops are not considered more than once, and (3) there are a total of $O(N^2)$ possible base pairs bounding extensible loops in a sequence of length N, the overall sampling algorithm scales as $O(N^2)$. This matches the asymptotic scaling of Ding and Lawrence[3], while including the complete class of large interior loops, some of which they exclude.

S4.6 Operation order for suboptimal structure generation

In many cases, the core features of a complex ensemble may be summarized by its MFE proxy structure(s) (2.10), $s_{\mathrm{MFE}'}$, or the set of all stacking states below a given free energy gap $\overline{\Gamma}_{\mathrm{subopt}}(\phi, \Delta G_{\mathrm{gap}})$ (2.11). The set $\overline{\Gamma}_{\mathrm{subopt}}(\phi, \Delta G_{\mathrm{gap}})$ can be equivalently viewed as the set of structures corresponding to stacking states $s^{\shortparallel}$ whose equilibrium probability $\overline{p}(\phi, s^{\shortparallel})$ is at least $\overline{p}_{\mathrm{gap}} \equiv \exp\{-[\overline{\Delta G}(\phi, s^{\shortparallel}_{\mathrm{MFE}}) + \Delta G_{\mathrm{gap}}]/kT\}/\overline{Q}(\phi)$. $\overline{\Gamma}_{\mathrm{subopt}}(\phi, \Delta G_{\mathrm{gap}})$ is just the MFE proxy structure(s), $s_{\mathrm{MFE}'}$, when $\Delta G_{\mathrm{gap}} = 0$, and algorithmically we therefore focus on calculation of $\overline{\Gamma}_{\mathrm{subopt}}(\phi, \Delta G_{\mathrm{gap}})$.

The program flow for determining suboptimal structures is controlled by a stack data structure containing partial structures $\{s\}$. Each partial structure s represents all structures consistent with a given set of elements that have energies below a free energy gap. It is defined as a tuple of (1) a priority queue of recursion elements, (2) a free energy, and (3) a list of base pairs.

Using Algorithm S8, structure generation proceeds by popping the highest priority element λ from the top partial structure s on the stack. The appropriate recursion for the element is used to iterate through the set of all alternate conditional ensemble contributions via the ArgMinGap evaluation algebra. As for sampling, the InteriorBacktrackIntra and InteriorBacktrackInter subroutines (Section S4.5) are used for the interior loop recursions. For each alternate contribution falling below the given free energy gap, a new partial structure s' is generated from s. If a given contribution contained no elements and the priority queue of s' is empty, s' is output as a complete structure; otherwise, s' is pushed on the stack. The algorithm begins by pushing a partial structure corresponding to $Q_{1,N}$ onto the stack and proceeds until the stack is empty.

Using a stack data structure, the algorithm runs in a depth first manner to discover completed structures as early as possible. This allows emitting completed structures in a streaming fashion while additional structures are determined. The algorithm yields $\overline{\Gamma}_{\mathrm{subopt}}(\phi, \Delta G_{\mathrm{gap}})$ from sequence ϕ of length $N \equiv |\phi|$ with time complexity $O(|L|N^2)$ for $|L|$ suboptimal structures within the specified energy gap. This bound reflects the worst-case of a set of $|L|$ structures that contain no common recursion elements. Each structure must then be independently backtracked, incurring the worst-case $O(N^2)$ complexity bound of Section S4.5. Because the number of structures returned, $|L|$, is sequence- and parameter-dependent and potentially exponential in N, we did not attempt to bound the time complexity further.

Algorithm S8: Operation order for suboptimal structure generation.

1. Initialize empty stack $\mathcal{S}$ of partial structures and empty multiset L of complete structures.
2. Create parent partial structure s containing just the element $\lambda = (\varnothing, 1, N)$ and push it onto $\mathcal{S}$.
3. While $\mathcal{S}$ is not empty:
   a) Pop the first partial structure s off of the stack $\mathcal{S}$.
   b) If there are no elements in s, it is complete, so add it to L and continue the while loop.
   c) Otherwise, dequeue the first element λ from s.
   d) Update the free energy of s via $s_{\text{energy}} \leftarrow s_{\text{energy}} - Q_{\text{MinSum}}(\lambda)$. $Q_{\text{MinSum}}(\lambda)$ is the matrix element value obtained from the forward pass MFE calculation.
   e) If λ denotes any element $Q^b_{i,j}$, add a base pair $i \cdot j$ in structure s.
   f) Calculate the generator $G = R_{\text{ArgMinGap}}(\lambda)(\kappa)$. Essentially, the generator achieves iteration over each possible conditional ensemble contribution to λ. For exposition, this may be achieved with the coroutine $\kappa(x, \Lambda)$ which yields (x, Λ) and returns κ. In practice, the loop below was programmatically implemented via a callback function.
   g) For each contribution (x, Λ) in G where $s_{\text{energy}} + x \leq \Delta G_{\text{gap}} + \overline{\Delta G}(\phi, s^{||}_{\text{MFE}})$:
      i. Initialize a new partial structure s' from s by copying the priority queue and list of base pairs from s and setting its free energy to $s'_{\text{energy}} = s_{\text{energy}} + x$.
      ii. For each element $\lambda' \in \Lambda$, enqueue λ' into the priority queue of partial structure s'.
      iii. Push s' onto the stack $\mathcal{S}$.
4. Return L.

S5 Distinguishability issues

For a complex of L strands, the ensemble $\overline{\Gamma}$ treats each strand as distinct while the ensemble Γ treats strands with the same sequence as indistinguishable. Both ensembles have conceptual utility as they provide different perspectives when examining the physical properties of a complex. In laboratory experiments, strands with the same sequence are typically indistinguishable, so calculations over ensemble Γ are crucial for comparison to experimental data (e.g., equilibrium secondary structure probabilities and equilibrium complex concentrations). On the other hand, calculations over ensemble $\overline{\Gamma}$ can sometimes provide information that is valuable precisely because it cannot be measured experimentally (e.g., equilibrium base-pairing probability matrix).

All of the dynamic programs described in the present work operate on ensemble $\overline{\Gamma}$ using free energy model (2.1) where each strand is treated as distinct. This is a matter of algorithmic necessity, as the free energy model (2.4) used for ensemble Γ contains a symmetry correction that depends on the global rotational symmetry R of each secondary structure $s \in \Gamma$. For efficiency reasons, the dynamic programs avoid explicitly enumerating each structure, instead operating on local loop free energies to incorporating information for multiple structures simultaneously while operating only on local loop free energies. As a result, the dynamic programs cannot incorporate a different global rotational symmetry correction for each structure because they never have access to global structural information. However, to facilitate comparisons to experimental data, physical quantities calculated using a dynamic program over ensemble $\overline{\Gamma}$ using physical model (2.1) can be post-processed to obtain the corresponding physical quantities over ensemble Γ using physical model (2.4). In the following sections, we outline the situation for each physical quantity treated in the present work.

S5.1 Partition function

The partition function dynamic program calculates $\overline{Q}(\phi) = Q_{1,N}$ (for a complex with N nucleotides) over ensemble $\overline{\Gamma}$ using free energy model (2.1) treating all strands as distinct. The Distinguishability Correction Theorem of Dirks et al. [7] shows that this quantity can be used to calculate the partition function $Q(\phi)$ over ensemble Γ using physical model (2.4) treating strands with the same sequence as indistinguishable. For convenience, we include the associated definitions and proof[7] here to enable extension of this analysis to other physical quantities.

Consider a complex of L strands with ordering π, where some of the strands may be indistinguishable. Let $\mathcal{G}$ be the group of $v(\pi)$ cyclic permutations mapping each strand to a strand of the same species. For example, $v(\pi) = 4$ for $\pi = AAAA$, $v(\pi) = 3$ for $\pi = ABABAB$, and $v(\pi) = 2$ for $\pi = ABAABA$, $v(\pi) = 1$ for $\pi =$ AAB, where the elements of $\mathcal{G}$ correspond to all rotations of a polymer graph that map strands of type $A \rightarrow A$ and strands of type $B \rightarrow B$. We term $v(\pi)$ the periodic strand repeat of the complex with ordering π.

For complexes in which all strands are distinct, $v(\pi) = 1$. Complexes containing multiple copies of the same strand species may have $v(\pi) > 1$, in which case the calculated partition function will be incorrect for ensemble Γ and free energy model (2.4) due to symmetry and redundancy errors that are different for different structures in the ensemble. For example, consider a complex with strand ordering $\pi = AAAA$ (Figure S36), that contains structures with either a 1-fold (i.e., no symmetry), 2-fold, or 4-fold rotational symmetry. Each of these cases will be treated incorrectly from the perspective of ensemble Γ and physical model (2.4). Dirks et al. [7] show that the symmetry and redundancy errors interact in such a way that they can be exactly and simultaneously corrected.

Consider an arbitrary secondary structure $s \in \overline{\Gamma}$. A permutation $g \in \mathcal{G}$ acts on a secondary structure s by relabeling strand identifiers: $g(s) = \{i_{g(m)} \cdot j_{g(n)} : i_m \cdot j_n \in s\}$. The stabilizer of s, $\mathcal{G}_s = \{g \in \mathcal{G} : g(s) = s\}$, is the set of cyclic permutations of strand identifiers (rotations of the polymer graph) that map s onto itself. The order of the rotational symmetry of the physical complex with secondary structure s is given by $|\mathcal{G}_s|$, requiring a correction of $+kT \log |\mathcal{G}_s|$ to the standard loop-based free energy.

The orbit of s in $\mathcal{G}$, $\mathcal{G}(s) = \{g(s) \in \overline{\Gamma} : g \in \mathcal{G}\}$, is the subset of $\overline{\Gamma}$ corresponding to the images of s under the permutations of the group $\mathcal{G}$. Note that the members of $\mathcal{G}(s)$ represent secondary structures within $\overline{\Gamma}$ that would be indistinguishable if the unique identifiers were removed from strands of the same species. Consequently, the partition function contribution of secondary structure $s \in \Gamma$ will be overcounted by a factor of $|\mathcal{G}(s)|$ because the dynamic program treats elements of the orbit as algorithmically distinct even though they are physically indistinguishable.

The orbit-stabilizer theorem of group theory [8] provides the useful relationship

$$|\mathcal{G}_s||\mathcal{G}(s)| = |\mathcal{G}| = v(\pi), \ \forall\, s \in \overline{\Gamma}$$

linking the symmetry and redundancy effects. Crucially, the product $|\mathcal{G}_s||\mathcal{G}(s)|$ depends only on the strand ordering π and is independent of the specific secondary structure $s \in \overline{\Gamma}$.

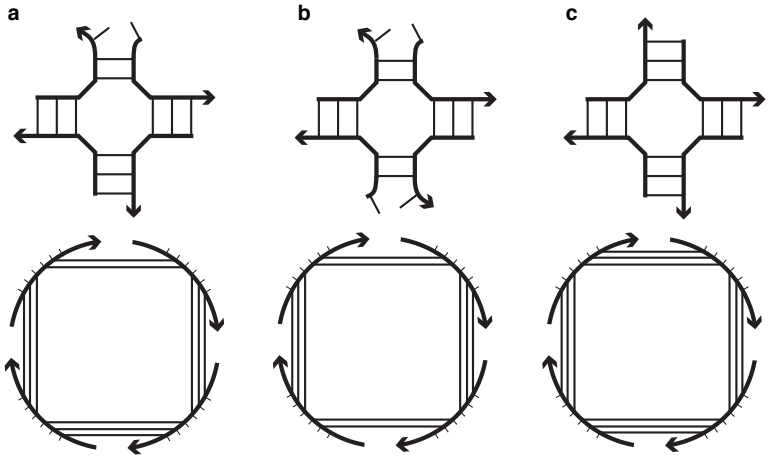


Figure S36: Example secondary structures and polymer graphs for a complex with strand ordering π =AAAA. The four strands have the same sequence and are distinct in ensemble $\overline{\Gamma}$ (each with a unique identifier in {1,2,3,4}) but indistinguishable in ensemble Γ. The partition function dynamic program operates on ensemble $\overline{\Gamma}$. After completing a calculation, if the strand identifiers are conceptually removed with the goal of converting the partition function $\overline{Q}(\phi)$ from ensemble $\overline{\Gamma}$ to the partition function $Q(\phi)$ in ensemble Γ, different structures in $\overline{\Gamma}$ have different rotational symmetries and different redundancies in Γ. Structures with an R-fold rotational symmetry are missing a penalty of $+kT \log R$ to the free energy model and hence are overweighted in the partition function by a factor of R. Structures with an S-fold redundancy are overcounted in the partition function by a factor of S. (a) 1-fold (i.e., no) rotational symmetry; 4-fold redundancy (4 indistinguishable structures as each stem plays the role of having 2 base pairs). (b) 2-fold rotational symmetry; 2-fold redundancy (2 indistinguishable structures as each opposing pair of stems plays the role of having 2 base pairs). (c) 4-fold rotational symmetry; 1-fold (i.e., no) redundancy.

Theorem S5.1 (Partition Function Distinguishability Correction)**.** *For a complex with strand ordering π, if the partition function dynamic program yields $\overline{Q}(\phi)$ for ensemble $\overline{\Gamma}$, then the partition function for ensemble Γ accounting for both symmetry and redundancy corrections is $Q(\phi) = \overline{Q}(\phi)/v(\pi)$.*

Proof. The partition function algorithm applied to ensemble $\overline{\Gamma}$ yields

$$\overline{Q}(\phi) = \sum_{s \in \overline{\Gamma}} \exp\{-\overline{\Delta G}(\phi, s)/kT\}. \tag{S91}$$

The partition function for ensemble Γ is then

$$\begin{aligned}
Q(\phi) &= \sum_{s \in \Gamma} \exp\{-\Delta G(\phi, s)/kT\} \\
&= \sum_{s \in \Gamma} \exp\{-(\overline{\Delta G}(\phi, s) + kT \log |\mathcal{G}_s|)/kT\} && \text{(S92)} \\
&= \sum_{s \in \Gamma} \sum_{\sigma \in \mathcal{G}(s)} \frac{1}{|\mathcal{G}(\sigma)|} \exp\{-(\overline{\Delta G}(\phi, \sigma) + kT \log |\mathcal{G}_\sigma|)/kT\} && \text{(S93)} \\
&= \sum_{s \in \overline{\Gamma}} \frac{1}{|\mathcal{G}(s)|} \exp\{-(\overline{\Delta G}(\phi, s) + kT \log |\mathcal{G}_s|)/kT\} && \text{(S94)} \\
&= \frac{1}{\nu(\pi)} \sum_{s \in \overline{\Gamma}} \exp\{-\overline{\Delta G}(\phi, s)/kT\} && \text{(S95)} \\
&= \frac{\overline{Q}(\phi)}{\nu(\pi)}. && \text{(S96)}
\end{aligned}$$

Thus, the symmetry and redundancy corrections combine to give a uniform factor $\nu(\pi)^{-1}$ that is independent of the structure $s \in \overline{\Gamma}$, enabling exact conversion of $\overline{Q}(\phi)$ into $Q(\phi)$. □

The partition function $Q(\phi)$ for ensemble Γ is suitable for calculating physical quantities that will be compared to experimental measurements in which strands of the same species are indistinguishable (e.g., equilibrium secondary structure probabilities $p(\phi, s)$ or equilibrium complex concentrations x). The corresponding complex free energy is

$$\Delta G(\phi) = -kT \log Q(\phi), \tag{S97}$$

which should not be confused with $\Delta G(\phi, s)$, the free energy of a single secondary structure $s \in \Gamma$.

S5.2 Equilibrium secondary structure probability

In ensemble $\overline{\Gamma}$ treating all strands as distinct, the equilibrium probability of any secondary structure $s \in \overline{\Gamma}$ is:

$$\overline{p}(\phi, s) = \frac{1}{\overline{Q}(\phi)} \exp\{-\overline{\Delta G}(\phi, s)/kT\} \tag{S98}$$

where $\overline{Q}(\phi)$ is the partition function over ensemble $\overline{\Gamma}$ treating all strands as distinct and $\overline{\Delta G}(\phi, s)$ is calculated using (2.1).

In ensemble Γ treating strands with the same sequence as indistinguishable, the equilibrium probability of any secondary structure $s \in \Gamma$ is:

$$p(\phi, s) = \frac{1}{Q(\phi)} \exp\{-\Delta G(\phi, s)/kT\} \tag{S99}$$

where $Q(\phi)$ is calculated using (S96) and $\Delta G(\phi, s)$ is calculated using (2.4).

The relationship between the probabilities in the two ensembles is given by:

$$\begin{aligned}
p(\phi, s) &= \frac{1}{Q(\phi)} \exp\{-\Delta G(\phi, s)/kT\} && \text{(S100)}\\
&= \frac{\nu(\pi)}{\overline{Q}(\phi)} \exp\{-[\overline{\Delta G}(\phi, s) + kT \log |\mathcal{G}_s|]/kT\} && \text{(S101)}\\
&= \frac{\nu(\pi)}{\overline{Q}(\phi)} \sum_{\sigma \in \mathcal{G}(s)} \frac{1}{|\mathcal{G}(\sigma)|} \exp\{-[\overline{\Delta G}(\phi, \sigma) + kT \log |\mathcal{G}_\sigma|]/kT\} && \text{(S102)}\\
&= \frac{1}{\overline{Q}(\phi)} \sum_{\sigma \in \mathcal{G}(s)} \exp\{-\overline{\Delta G}(\phi, \sigma)/kT\} && \text{(S103)}\\
&= \sum_{\sigma \in \mathcal{G}(s)} \overline{p}(\phi, \sigma) && \text{(S104)}\\
& && \text{(S105)}
\end{aligned}$$

where the structures in the set $\mathcal{G}(s)$ for $s \in \overline{\Gamma}$ become redundant if the distinct identifiers are removed from strands of the same species. Hence, $p(\phi, s)$ is the sum of the (identical) probabilities $\overline{p}(\phi, s)$ of these redundant structures.

S5.3 Equilibrium base-pairing probabilities

Using a bactrack-free dynamic program, the matrix of equilibrium base-pairing probabilities $P(\phi)$ is calculated over ensemble $\overline{\Gamma}$ using free energy model (2.1) treating all strands as distinct. One may visualize a thought experiment in which all strands, all nucleotides, and all base-pairs are distinct, with equilibrium base-pairing probabilities available for each of these distinct base pairs. The probabilities in this matrix are not directly comparable to experimental measurements in which strands of the same sequence are indistinguishable, but nonetheless provide a valuable and detailed window into the complex ensemble.

Let

$$p(i_1 \cdot j_2) \tag{S106}$$

denote the equilibrium probability for base-pair $(i_1 \cdot j_2)$ with nucleotide i of a strand with identifier 1 pairing to nucleotide j of a strand with identifier 2. Let

$$p(i_1) \tag{S107}$$

denote the equilibrium probability that base i of a strand with identifier 1 is unpaired.

S5.4 Structure sampling

The simultaneous sampling algorithm Boltzmann samples a set of J secondary structures

$$\overline{\Gamma}_{\text{sample}}(\phi, J) \tag{S108}$$

from ensemble $\overline{\Gamma}$ using free energy model (2.1) treating all strands as distinct. Unlike the equilibrium base-pairing probability matrix $P(\phi)$, by averaging or clustering the sampled structures, it is possible to examine correlations between base pairs. As the number of sampled structures increases, the average structural properties over the sampled set recover the equilibrium base-pairing probability matrix:

$$P(\phi) = \lim_{J \to \infty} \frac{1}{J} \sum_{s \in \overline{\Gamma}_{\text{sample}}} S(s). \tag{S109}$$

A set of J structures $\overline{\Gamma}_{\text{sample}}(\phi, J)$ sampled from ensemble $\overline{\Gamma}$ where all strands are distinct can be post-processed to generate a set of structures $\Gamma_{\text{sample}}(\phi, J)$ sampled from ensemble Γ where strands with the same sequence are indistinguishable.

For ensemble $\overline{\Gamma}$ with free energy model (2.1), a structure $s \in \overline{\Gamma}$ is Boltzmann sampled with probability $\overline{p}(\phi, s)$ by the sampling dynamic program, yielding an integer number of samples $\overline{n}_{\text{sample}}(\phi, s) \in \{0, \ldots, J\}$. Conceptually, for ensemble Γ with free energy model (2.4), a structure $s \in \Gamma$ would be Boltzmann sampled with probability $p(\phi, s)$. We have previously derived the relationship (S104) between the equilibrium probabilities in the two ensembles:

$$p(\phi, s) = \sum_{\sigma \in \mathcal{G}(s)} \overline{p}(\phi, \sigma). \tag{S110}$$

The equilibrium probability of a structure $s \in \Gamma$ is simply the sum of the equilibrium probabilities of the structures $\sigma \in \mathcal{G}(s)$ that are indistinguishable in ensemble $\overline{\Gamma}$ upon removal of their unique identifiers. Hence, the sample count for Boltzmann sampling from ensemble Γ with free energy model (2.1) is obtained by summing the sample counts for the

structures $\sigma \in \mathcal{G}(s)$ that are indistinguishable in ensemble $\overline{\Gamma}$ upon removal of their unique identifiers:

$$n_{\text{sample}}(\phi, s) = \sum_{\sigma \in \mathcal{G}(s)} \overline{n}_{\text{sample}}(\phi, \sigma). \tag{S111}$$

S5.5 Equilibrium complex concentrations

Consider a test tube ensemble containing a set of strand species Ψ^0 interacting to form the set of complex species Ψ. To calculate the set of equilibrium concentrations $x_\Psi \equiv x_c \, \forall c \in \Psi$, we first calculate the set of partition functions Q_Ψ using (S96). The complex concentrations x_Ψ (specified as mole fractions) are then the unique solution to the strictly convex optimization problem [7]:

$$\min_{x_\Psi} \sum_{c \in \Psi} x_c (\log x_c - \log Q_c - 1) \tag{S112a}$$

$$\text{subject to} \quad \sum_{c \in \Psi} A_{i,c} x_c = x_i^0 \quad \forall i \in \Psi^0. \tag{S112b}$$

Here, A is the stoichiometry matrix such that $A_{i,c}$ is the number of strands of type i in complex c and x_i^0 denotes the total concentration of strand species i in the test tube. Following Dirks et al., [7], this problem is solved efficiently in the dual form as an unconstrained convex optimization problem using a trust-region method with a Newton dog-leg step [16] using Cholesky decomposition for the Newton matrix inversions.

S5.6 Ensemble pair fractions

If a complex contains some indistinguishable strands, distinguishability effects arise at the secondary structure level in the form of rotational symmetry corrections and algorithmic overcounting corrections (Section S5.1). New distinguishability issues arise when examining individual base pairs within these secondary structures [7]. For example, consider a complex $\pi = AAB$ involving two indistinguishable copies of strand A (with identifiers 1 and 2) and one copy of strand B (with identifier 3). Periodic strand repeat $\nu(\pi) = 1$ so no symmetry and overcounting corrections are required for any structure $s \in \Gamma$. However, base pairs $(i_1 \cdot j_3)$ and $(i_2 \cdot j_3)$ are indistinguishable since strands 1 and 2 are both of type A. Likewise, without the global structural context, the inter- and intra-strand base pairs $(i_1 \cdot j_2)$ and $(i_1 \cdot j_1)$ are also indistinguishable. Fortunately, the equilibrium base-pairing probabilities calculated over ensemble Γ (Section S5.3) can be used to calculate base-pairing observables that account for this indistinguishability.

First, consider a complex in which strands with the same sequence are indistinguishable. Let Θ be the set of strand species in the complex and $\{\theta\}$ be the set of all strand identifiers corresponding to strands of type $\theta \in \Theta$ (hence $L = \sum_{\theta\in\Theta} \left|\{\theta\}\right|$). We define the expected number of base pairs between base i on strands of type $A \in \Theta$ and base j on strands of type $B \in \Theta$ to be $E(i_{\{A\}} \cdot j_{\{B\}}) \in [0, \min(|\{A\}|, |\{B\}|)]$. For a given complex,

$$E(i_{\{A\}} \cdot j_{\{B\}}) = \sum_{l_A\in\{A\}} \sum_{l_B\in\{B\}} p(i_{l_A} \cdot j_{l_B})$$

represents a sum over the contributions of each type of distinct base pair, where each term $p(i_{l_A} \cdot j_{l_B})$ is an equilibrium base-pairing probability (S106).

Now consider a test tube in which strands with the same sequence are indistinguishable. Let Ψ^0 denote the set of strand species that interact to form the set of complex species Ψ. For a complex $k \in \Psi$, let $E_k(i_{\{A\}} \cdot j_{\{B\}})$ denote the expectation value that base i of strand species $A \in \Theta_k$ pairs to base j of strand species $B \in \Theta_k$, where $\Theta_k \subseteq \Psi^0$ denotes the set of strand species that appear in complex k. For a test tube ensemble at equilibrium, the expected concentration of base pairs between base i of strands of type A and base j of strands of type B is

$$x(i_A \cdot j_B) = \sum_{k\in\Psi} x_k E_k(i_{\{A\}} \cdot j_{\{B\}}).$$

For experimental studies, it is usually more convenient to measure the expected fraction of A strands or B strands that form this base pair:

$$f_A(i_A \cdot j_B) = x(i_A \cdot j_B)/x_A^0 \tag{S113}$$

$$f_B(i_A \cdot j_B) = x(i_A \cdot j_B)/x_B^0, \tag{S114}$$

respectively. These ensemble pair fractions are conceptually suitable for comparison to a FRET experiment designed to measure formation of a base-pair between base i of strands of type A with base j of strands of type B.

Similarly, the concentration $x(i_A)$ of strand species $A \in \Psi^0$ with base i unpaired is

$$x(i_A) = x_A^0 - \sum_{B\in\Psi^0} \sum_{j=1}^{N_B} x(i_A \cdot j_B),$$

and the fraction of A strands that have base i unpaired is

$$f_A(i_A) = x(i_A)/x_A^0. \tag{S115}$$

The total concentration of unpaired bases in solution is

$$x_{\text{unpaired}} = \sum_{A\in\Psi^0} x(i_A) = \sum_{k\in\Psi} x_k \sum_{j=1}^{N_k} P^{j,j}(\phi_k) \tag{S116}$$

and the total fraction of unpaired bases in solution is

$$f_{\text{unpaired}} = x_{\text{unpaired}} / \sum_{A \in \Psi^0} x_A^0 N_A. \tag{S117}$$

The total fraction unpaired is conceptually suitable for comparison to an absorbance measurement.

S5.7 MFE free energy and secondary structure

The MFE dynamic program returns the free energy of the MFE stacking state in ensemble $\overline{\Gamma}$ using free energy model (2.1):

$$\overline{\Delta G}(\phi, s^{\shortparallel}_{\text{MFE}}). \tag{S118}$$

Note that the MFE algorithm does not return the free energy of the MFE secondary structure s_{MFE} but rather the free energy of the MFE stacking state $s^{\shortparallel}_{\text{MFE}}$. This is a consequence of the recursions operating on stacking state as the elementary state. The backtracking dynamic program then returns the secondary structure

$$s_{\text{MFE}'} = \{s \in \overline{\Gamma} | s^{\shortparallel}_{\text{MFE}} \in s, s^{\shortparallel}_{\text{MFE}}(\phi) = \arg\min_{s^{\shortparallel} \in \overline{\Gamma}^{\shortparallel}} \overline{\Delta G}(\phi, s^{\shortparallel})\}. \tag{S119}$$

that contains $s^{\shortparallel}_{\text{MFE}}$ within its subensemble. Thus, this structure is not the MFE secondary structure, s_{MFE}, but rather a proxy $s_{\text{MFE}'}$ that contains $s^{\shortparallel}_{\text{MFE}}$ within its subensemble. The free energy of this secondary structure can be cheaply evaluated in ensemble $\overline{\Gamma}$ using (2.1) to yield $\overline{\Delta G}(\phi, s_{\text{MFE}'})$ or in ensemble Γ using (2.4) to yield $\Delta G(\phi, s_{\text{MFE}'})$.

Because the recursions operate on stacking states as the elementary state, it is not clear how to calculate the MFE free energy $\overline{\Delta G}(\phi, s_{\text{MFE}})$ and secondary structure s_{MFE} for ensemble $\overline{\Gamma}$. As a result, there is also no starting point for post-processing these results to calculate $\Delta G(\phi, s_{\text{MFE}})$ or s_{MFE} for ensemble Γ.

This situation is not entirely satisfactory. By definition, an MFE secondary structure has the highest equilibrium probability, $\overline{p}(\phi, s_{\text{MFE}})$, in structural ensemble $\overline{\Gamma}$. However, $\overline{p}(\phi, s_{\text{MFE}})$ can nonetheless be arbitrarily small due to competition from other structures in $\overline{\Gamma}$. For ensembles where $p(\phi, s_{\text{MFE}})$ is non-negligible, an attractive alternative to the deterministic approach is to use Boltzmann sampling to discover the MFE secondary structure. One advantage of the random approach is that it determines MFE status based on secondary structure s rather than subensemble stacking state $s^{\shortparallel} \in s$.

Sampling is performed for ensemble $\overline{\Gamma}$ treating all strands as distinct. Suppose that the identity of s_{MFE} is unknown, as is its free energy $\overline{\Delta G}(\phi, s_{\text{MFE}})$ and its equilibrium probability

$\overline{p}(\phi, s_{\mathrm{MFE}})$. The probability, p_{fail}, that a sample of J structures does not include a structure s_{MFE} that has probability $\overline{p}(\phi, s_{\mathrm{MFE}}) \geq p_{\min}$ is

$$p_{\mathrm{fail}} \leq (1 - p_{\min})^J. \tag{S120}$$

Inverting this relationship, for a given $p_{\min}$, we can calculate the number of samples, J, required to assure a failure probability no higher than p_{fail}:

$$J \geq \frac{\log p_{\mathrm{fail}}}{\log(1 - p_{\min})} \approx \frac{\log p_{\mathrm{fail}}}{-p_{\min}}. \tag{S121}$$

Because the dependence of J on p_{fail} is logarithmic, it is inexpensive to reduce p_{fail} for fixed $p_{\min}$. For example, for $p_{\min} = 0.01$, we have $J \geq 688$ for $p_{\mathrm{fail}} = 10^{-3}$ and $J \geq 2750$ for $p_{\mathrm{fail}} = 10^{-12}$. However, the required number of samples is sensitive to the value of $p_{\min}$ (the assumed lower bound in the MFE probability). For example, holding $p_{\mathrm{fail}} = 10^{-12}$ fixed, we require $J \geq 27,618$ samples for $p_{\min} = 0.001$ and $J \geq 276,297$ samples for $p_{\min} = 0.0001$. While that number of samples remains affordable using the new simultaneous sampling algorithm (Figure 2.15), if the MFE probability becomes vanishingly small, the required number of samples would grow too large to be practical. On the other hand, if the MFE probability is vanishingly small, the MFE structure may not provide a useful summary of the equilibrium base-pairing properties of the ensemble (in which case the equilibrium base-pairing probability matrix $P(\phi)$ will continue to provide such a summary).

After sampling J structures using the new simultaneous sampling method, let p_{MFE^*} denote the highest probability of the sampled structures, and let s_{MFE^*} denote the MFE proxy structure determined by random sampling. The probability that there exists an (undiscovered) MFE structure with $\overline{p}_{\mathrm{MFE}} \geq \overline{p}_{\mathrm{MFE}^*}$ is bounded by

$$p_{\mathrm{fail}} \leq [1 - \overline{p}(\phi, s_{\mathrm{MFE}^*})]^J. \tag{S122}$$

Hence, after sampling J structures, it is straightforward calculate the probability that the true MFE structure was not identified. If desired, additional samples can be performed to increase J (potentially identifying a higher p_{MFE^*}) and further reduce the failure rate.

One of the other drawbacks of the deterministic approach of equations (2.9) and (2.10) is that it does not treat ensemble Γ where strands with the same sequence are indistinguishable, which is the circumstance for typical experimental measurements. However, the random MFE algorithm can be applied using samples from ensemble Γ, in which case the a posteriori failure bound (S122) is replaced by

$$p_{\mathrm{fail}} \leq [1 - p(\phi, s_{\mathrm{MFE}^*})]^J. \tag{S123}$$

The MFE free energy $\Delta G(\phi, s_{\text{MFE}^*})$ is then directly comparable to the complex free energy $\Delta G(\phi)$ (S97) for ensemble Γ with

$$\Delta G(\phi) \leq \Delta G(\phi, s_{\text{MFE}^*}). \tag{S124}$$

See Section S6.7 for an empirical comparison of deterministic and random algorithms in calculating the MFE free energy and secondary structure.

S6 Additional studies

Except where otherwise noted, computational studies were performed for ensemble `stacking` with parameters `rna95` (Section S1.5) for RNA at 37 °C in 1 M Na^+, subject to two historical modifications: 1) `G·U` wobble pairs are prohibited as terminal base pairs in exterior loops and multiloops, and 2) terminal mismatch free energies are replaced by two dangle stacking free energies in exterior loops and multiloops (see equation (S55). All benchmarks were run on AWS EC2 C5 instances using a single computational core (3.0 GHz Intel Xeon Platinum processors with 72 GB of memory, except 144 GB for of memory for figures involving complexes containing 30,000 nt).

S6.1 Comparison of predictions to structure databases

One approach to evaluating the quality of secondary structure models is make predictions for databases of structures drawn from comparative sequence analysis and/or tertiary structure measurements [1, 12]. Here, were compare the complex ensembles `nostacking` and `stacking` for two RNA parameter sets `rna95` and `rna06` using the "Archive II" structure database from Reference [12] and the "SSTRAND" structure database from Reference [1]. The free energy parameters in the models we are testing were regressed based on experiments in 1M Na^+. By contrast, the database structures reflect a range of experimental conditions. As a result, it is unclear whether improvements in the free energy model (loop free energy parameter sets) and/or the structural ensemble (stacking/no stacking) in modeling RNA in 1M Na^+ should be expected to yield convergence to database reference structures. For this reason, we draw no conclusions, but nonetheless document comparisons between predictions and database structures to serve as a reference (Table A.2). We calculated three quantities:

- the normalized complex ensemble defect [6, 25]

$$1 - N^{-1} \sum_{\substack{1 \le i \le N \\ 1 \le j \le N}} \overline{P}^{i,j}(\phi)\, S^{i,j}(s_*) \quad \in (0, 1)$$

  representing the equilibrium fraction of nucleotides that are paired differently over the complex ensemble relative to the database structure. Here, N is the number of nucleotides, $\overline{P}(\phi)$ is the calculated base-pairing probability matrix, and $S(s_*)$ is the structure matrix corresponding to the database structure s_*.

- the normalized MFE defect [25]

$$1 - N^{-1} \sum_{\substack{1 \le i \le N \\ 1 \le j \le N}} S^{i,j}(s_{\mathrm{MFE}'})\, S^{i,j}(s_*) \quad \in [0,1]$$

  representing the fraction of nucleotides in the MFE proxy structure $s_{\mathrm{MFE}'}$ that are paired differently relative to the database structure s_*.

- the F-measure [12]

$$\frac{2qr}{q+r} \quad \in [0,1]$$

  representing the harmonic mean of the precision q (the fraction of pairs in the predicted $s_{\mathrm{MFE}'}$ that are in the database structure s_*) and the sensitivity r (the fraction of pairs in the database structure s_* that are in the predicted $s_{\mathrm{MFE}'}$).

Note that at equilibrium, a sequence will adopt an ensemble of secondary structures. However, the structure database typically records only a single structure per sequence. The MFE defect and F-measure similarly represent the computed equilibrium structural ensemble using a single MFE proxy structure $s_{\mathrm{MFE}'}$. Hence, these two quantities are comparing one representative structure to another, potentially neglecting non-negligible contributions by other structures in the ensemble.

a

Parameters	Structural ensemble	Ensemble defect	MFE defect	F-measure
`rna06`	`nostacking`	0.492	0.479	0.476
	`stacking`	0.450	0.448	0.525
`rna95`	`nostacking`	0.393	0.372	0.598
	`stacking`	0.422	0.393	0.568

b

Parameters	Structural ensemble	Ensemble defect	MFE defect	F-measure
`rna06`	`nostacking`	0.463	0.452	0.479
	`stacking`	0.433	0.427	0.513
`rna95`	`nostacking`	0.406	0.392	0.556
	`stacking`	0.386	0.370	0.583

Table A.2: Comparison of predictions to structure databases. (a) "Archive II" database from Reference [12] (excluding pseudoknotted structures). (b) "SSTRAND" database from Reference [1]. Calculations performed for RNA at 37°C in 1 M Na^+ using either the `rna95` or `rna06` parameter sets with either the `nostacking` or `stacking` structural ensemble. Ensemble defect and MFE defect approach 0 as predictions approach a database structure. F-measure approaches 1 as predictions approach a database structure.

S6.2 Empirical dependence of ensemble size on complex size

We performed calculations to measure the number of secondary structures, $|\overline{\Gamma}(\phi)|$, and stacking states, $|\overline{\Gamma}^{\shortparallel}(\phi)|$, for a set of complexes with random sequences. Empirically, $|\overline{\Gamma}(\phi)|$ and $|\overline{\Gamma}^{\shortparallel}(\phi)|$ grow exponentially with the number of nucleotides in the complex (Figure S37). Least-squares linear regressions on the log-linear data yielded the fits $|\overline{\Gamma}(\phi)| = 0.00156 \cdot 1.770^N$ ($r = 0.9999994$) and $|\overline{\Gamma}^{\shortparallel}(\phi)| = 0.00650 \cdot 2.023^N$ ($r = 0.9999996$). Note that these results are sequence-dependent (e.g., $\phi =$ `AAAAA...` will have an ensemble size of $|\overline{\Gamma}(\phi)| = 1$ independent of complex size).

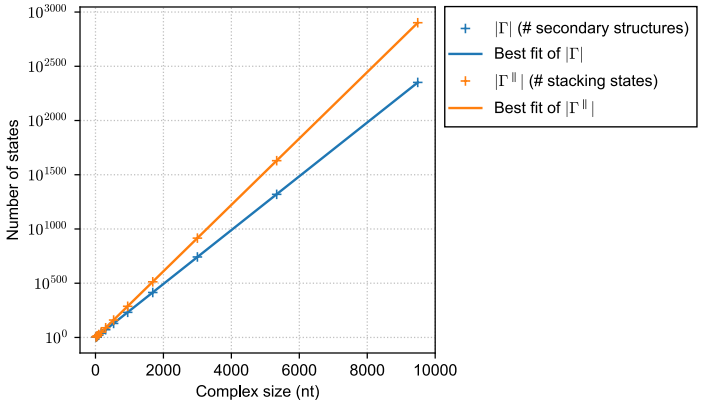


Figure S37: Empirical dependence of ensemble size on complex size. Each complex comprises 3 random RNA strands of equal length. Each data point represents the mean over 10 replicates with different sequences.

S6.3 Empirical dependence of partition function on complex size

We use our overflow-safe partition function algorithm (with evaluation algebra SplitExp) to calculate partition functions of both random complexes of 3 strands and designed duplexes to determine for what complex sizes overflow is predicted to occur using a non-overflow-safe partition function algorithm (corresponding to evaluation algebra SumProduct) with different floating point formats (single, double, and quad precision). For the designed duplexes, NUPACK was used to reduce the complex ensemble defect below 1% [23, 25]. Relative to random sequences, the designed sequences result in a deeper well on the free energy landscape and a larger partition function for a given complex size (Figure S38). Least-squares linear regression of log-linear data yielded the fits: $\log Q = 0.5146N - 7.305$ ($r = 0.999986$) for random complexes and $\log Q = 1.5614N - 4.266$ ($r = 0.999993$) for designed duplexes. Based on the maximum representable values with different floating point formats, random complexes are predicted to overflow at 187 nt, 1,393 nt, and 22,080 nt with single, double, and quad precision, respectively. The designed duplexes were predicted to overflow at 58 nt, 456 nt, and 7,275 nt with single, double, and quad precision, respectively. Without the overflow-safe evaluation algebra, edge-case sequences such as the repeating sequence ϕ = `GGG...CCC...` have been observed to cause overflow at sequence sizes as low as 4,500 nt using quadruple precision. Figure S39 displays example MFE proxy structures for random and designed sequences; the designed sequence has a larger partition function with nucleotides that adopt the depicted base-pairing state with higher probability on average.

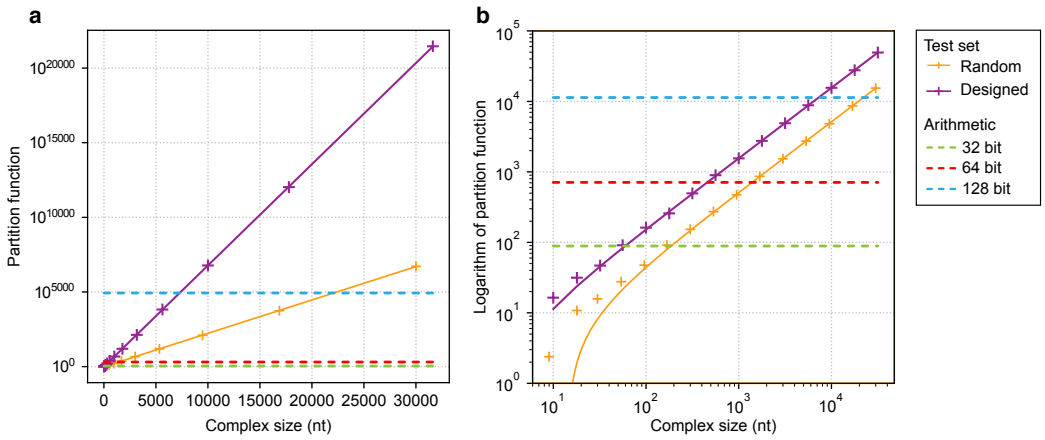


Figure S38: Dependence of partition function on complex size for random and designed sequences. (a) Partition function on a log scale vs complex size on a linear scale (log-linear data). (b) Same data plotted as log of the partition function on a log scale vs complex size on a log scale (loglog-log data). The thresholds for overflow using different float point formats are plotted as dashed lines, demonstrating that the overflow-safe algebra enables calculations for larger complexes. Solid lines of best fit in panel (a) are plotted as solid curves in panel (b). Each random complex comprises 3 RNA strands of equal length. Each designed duplex comprises 2 RNA strands of equal length. Each data point represents a mean over 5 replicate sequences.

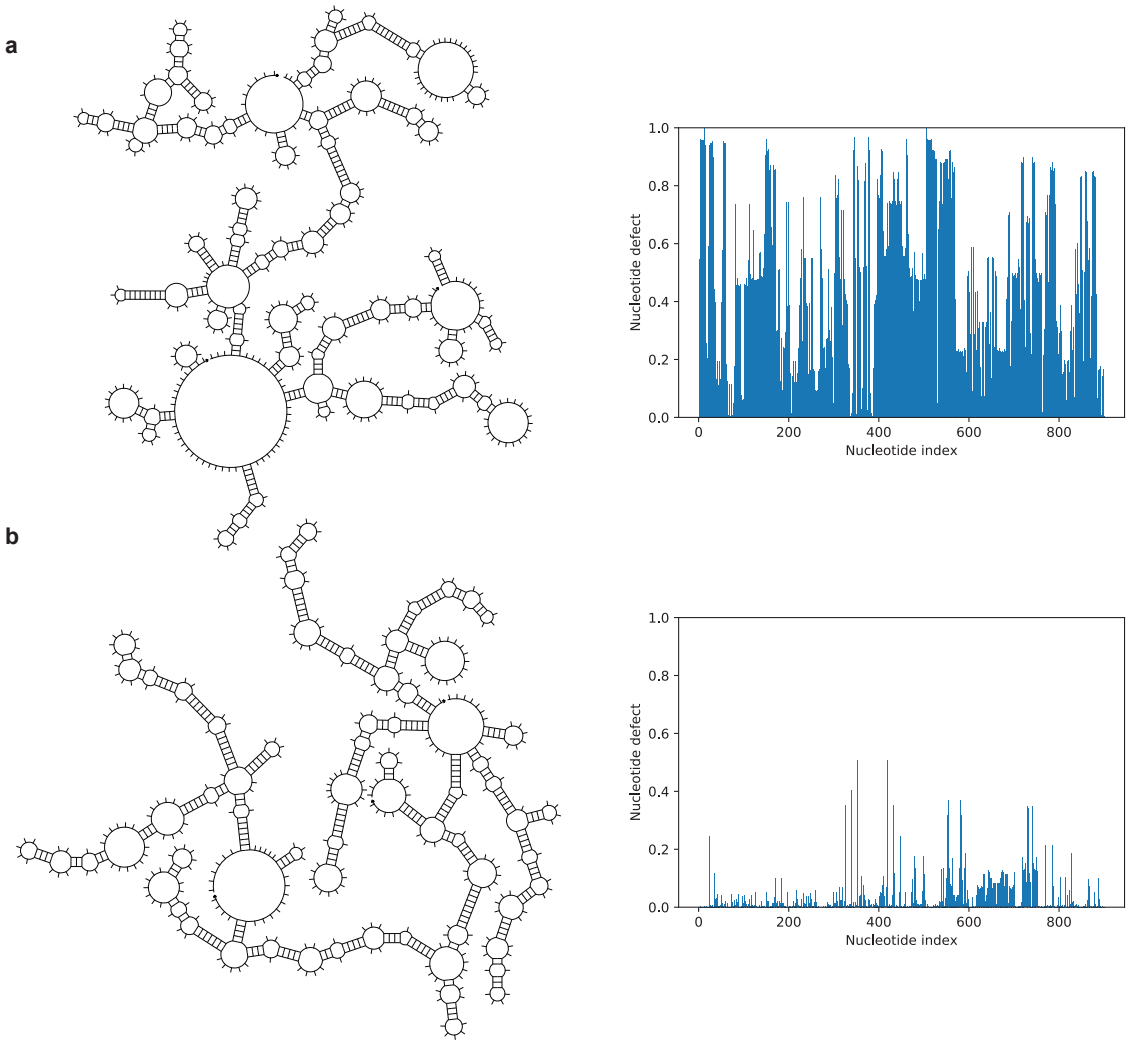


Figure S39: Example MFE proxy structures for random and designed sequences. a) Random sequence. Left: MFE proxy structure $s_{\mathrm{MFE}'}$ for a 3 × 300 nt trimer, partition function $Q(\phi) = 1.537 \cdot 10^{187}$, complex free energy $\Delta G(\phi) = -265.6$ kcal/mol. Right: nucleotide defect with respect to $s_{\mathrm{MFE}'}$. b) Designed sequence: MFE proxy structure $s_{\mathrm{MFE}'}$ for a 3 × 300 nt trimer, partition function $Q(\phi) = 2.092 \cdot 10^{279}$, complex free energy $\Delta G(\phi) = -396.4$ kcal/mol. Right: nucleotide defect with respect to $s_{\mathrm{MFE}'}$. Nucleotide defect relative to $s_{\mathrm{MFE}'}$ represents the probability that a given nucleotide does not adopt the depicted MFE base-pairing state.

S6.4 Relative cost of partition function, equilibrium pair probability, and MFE calculations

The cost of calculating the partition function, $\overline{Q}(\phi)$, equilibrium pair probability matrix, $\overline{P}(\phi)$, and MFE, $\overline{\Delta G}(\phi, s^{||}_{\text{MFE}})$ are profiled using $O(N^3)$ dynamic programs in Figure S40. Relative to the partition function, the MFE algorithm is comparable for small complexes and up to $\approx 4\times$ faster for large complexes due to the use of the overflow-safe evaluation algebra for the partition function. The pair probabilities algorithm is roughly $2\times$ the cost of the partition function algorithm, except for a spike to $\approx 3\times$ at complex sizes around the threshold for when the overflow-safe evaluation algebra turns on for the pair probabilities algorithm.

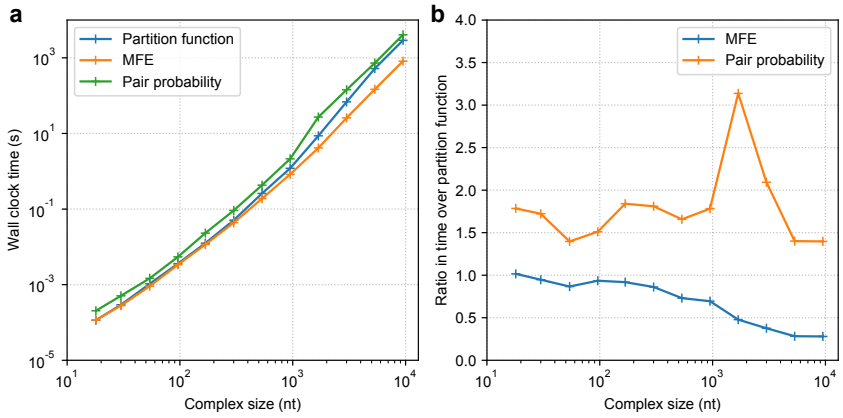


Figure S40: Relative cost of partition function, equilibrium pair probability, and MFE calculations. (a) Computational cost. (b) Relative cost of MFE and pair probability calculations to partition function calculations. Each complex comprises 3 RNA strands with random sequences of equal length. Each data point represents a mean over 5 replicate sequences.

S6.5 Speed and scalability of partition function calculations with different floating point formats and evaluation algebras

Figure S41 demonstrates the relative cost of performing partition function calculations with single-precision or double-precision floats using the non-overflow-safe SumProduct evaluation algebra, the overflow-safe SplitExp evaluation algebra, and the overflow-safe production implementation that dynamically switches from single-precision, to double-precision, to overflow-safe as required by the calculation. The overflow-safe evaluation algebra is roughly 2× slower than single-precision and double-precision floats, but enables calculations for larger complexes. The production approach transitions between the different costs as the complex size increases.

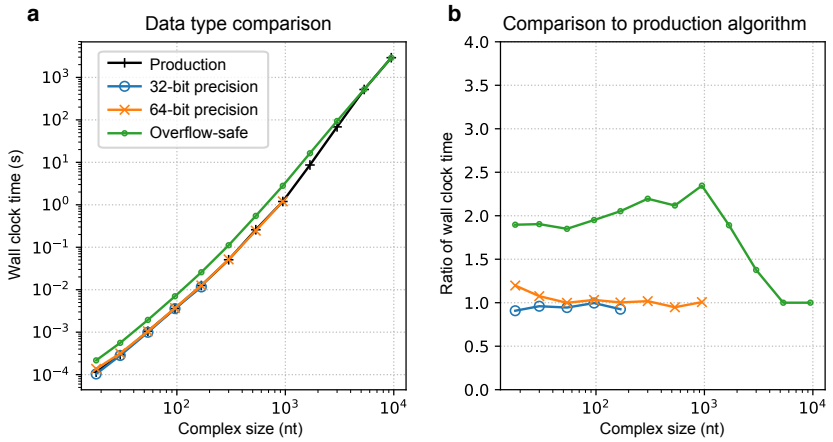


Figure S41: Speed and scalability of partition function calculations with different floating point formats and evaluation algebras. (a) Computational cost. (b) Cost relative to the production algorithm. Each complex comprises 3 RNA strands with random sequences of equal length. Each data point represents a mean over 5 replicate sequences.

S6.6 Performance of simultaneous vs sequential structure sampling

Here, we compare the cost of sequential vs simultaneous Boltzmann-sampling a set of J structures from a complex ensemble. The reported computation times do not include calculation of the partition function, which must precede structure sampling. We performed studies with complexes of either random sequences (Section S6.6.1) or designed sequences (Section S6.6.2) and then estimated the empirical algorithm complexities (Section S6.6.3). Speedups using simultaneous sampling are expected to increase for free energy landscapes characterized by a deep well due to the avoidance of sequentially resampling the same structural elements from the well. Hence, we would expect the typical speedup for designed sequences to be greater than for random sequences. For random sequences, the speedup using simultaneous vs sequential sampling is 7-10× for $J = 10^3$ samples and 10-24× for $J = 10^4$ samples. For designed sequences, the speedup using simultaneous vs sequential sampling is 9-34× for $J = 10^3$ samples and 11-55× for $J = 10^4$ samples. Because these designed complexes were generated using the MFE proxy structures of random sequences as the target structure for sequence design, they do not have particularly deep free energy wells compared to typical designed sequences (for example, these target structures will contain duplexes as short as 1 bp). As a result, the relative performance for simultaneous vs sequential sampling should increase even more for typical engineered complexes.

S6.6.1 Structure sampling for random complexes

Each random complex comprises 3 RNA strands with random sequences of equal length (ranging from 10 to 3000 nt each). Ten sets of replicate sequences were used for each complex size. For each complex, J structures were sampled (ranging from 10^1 to 10^6 structures). The computational cost of sampling J structures is plotted for each replicate in Figure S42. The mean speedup using simultaneous vs sequential sampling is plotted across complex sizes in Figure S43 and across number of samples in Figure S44.

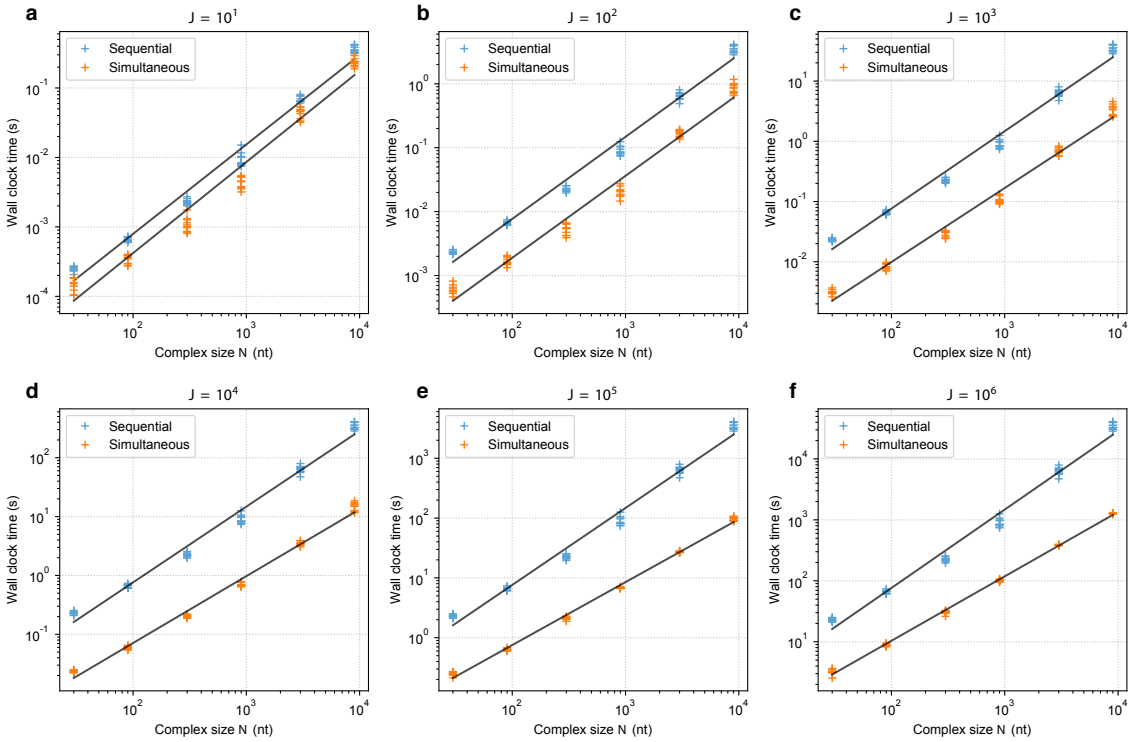


Figure S42: Cost of simultaneous and sequential structure sampling for random complexes for different numbers of samples $J \in 10^1, \ldots, 10^6$. J structures are sampled for each of 10 replicate sets of sequences per complex size. Each data point represents the sampling time for one replicate. Lines represent univariate regressions (see Section S6.6.3).

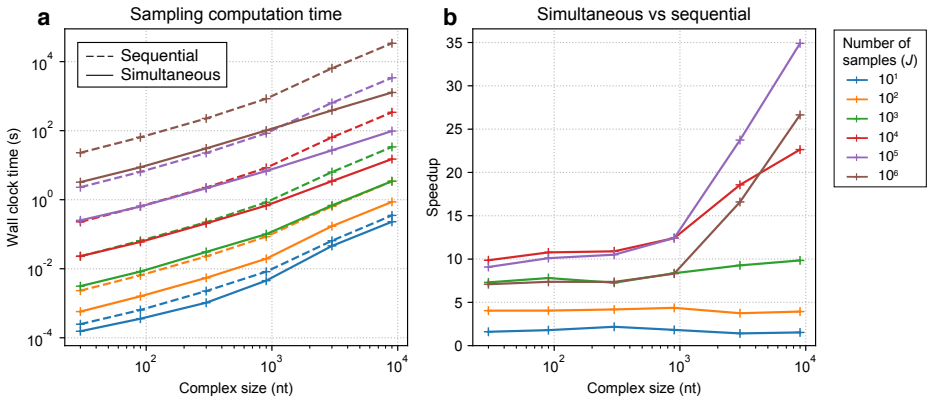


Figure S43: Sampling cost as a function of complex size (N) for random complexes. (a) Cost of simultaneous and sequential sampling. Each data point represents the mean over all replicates for a given complex size. (b) Speedup using simultaneous vs sequential sampling. Each data point represents the ratio of means.

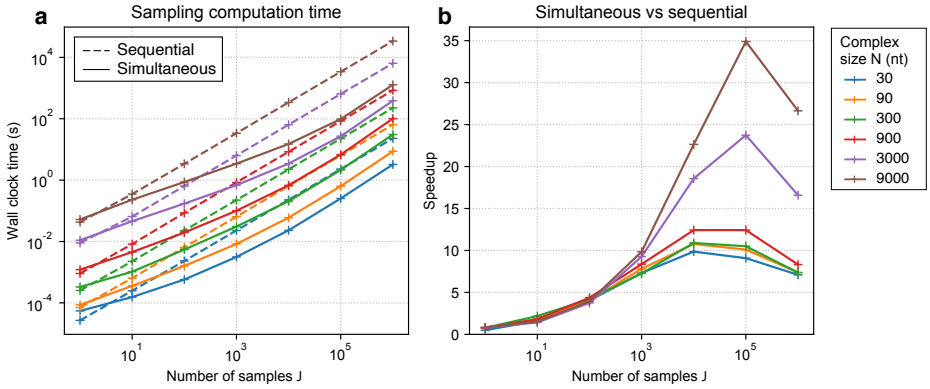


Figure S44: Sampling cost as a function of number of samples (J) for random complexes. (a) Cost of simultaneous and sequential sampling. Each data point represents the mean over all replicates for a given complex size. (b) Speedup using simultaneous vs sequential sampling. Each data point represents the ratio of means. Not that for sufficiently large J, the cost of sorting becomes significant, lessening the speedup of the simultaneous approach.

S6.6.2 Structure sampling for designed complexes

Each designed complex comprises 3 RNA strands with sequences of equal length (ranging from 10 to 3000 nt each). The sequences for a given designed complex were obtained by calculating the MFE proxy structure for a random complex, and then using that as the target structure for sequence design and reducing the complex ensemble defect below 1% [23, 25]. Ten sets of replicate sequences were used for each complex size. For each complex, J structures were sampled (ranging from 10^1 to 10^6 structures). The computational cost of sampling J structures is plotted for each replicate in Figure S45. The mean speedup using simultaneous vs sequential sampling is plotted across complex sizes in Figure S46 and across number of samples in Figure S47.

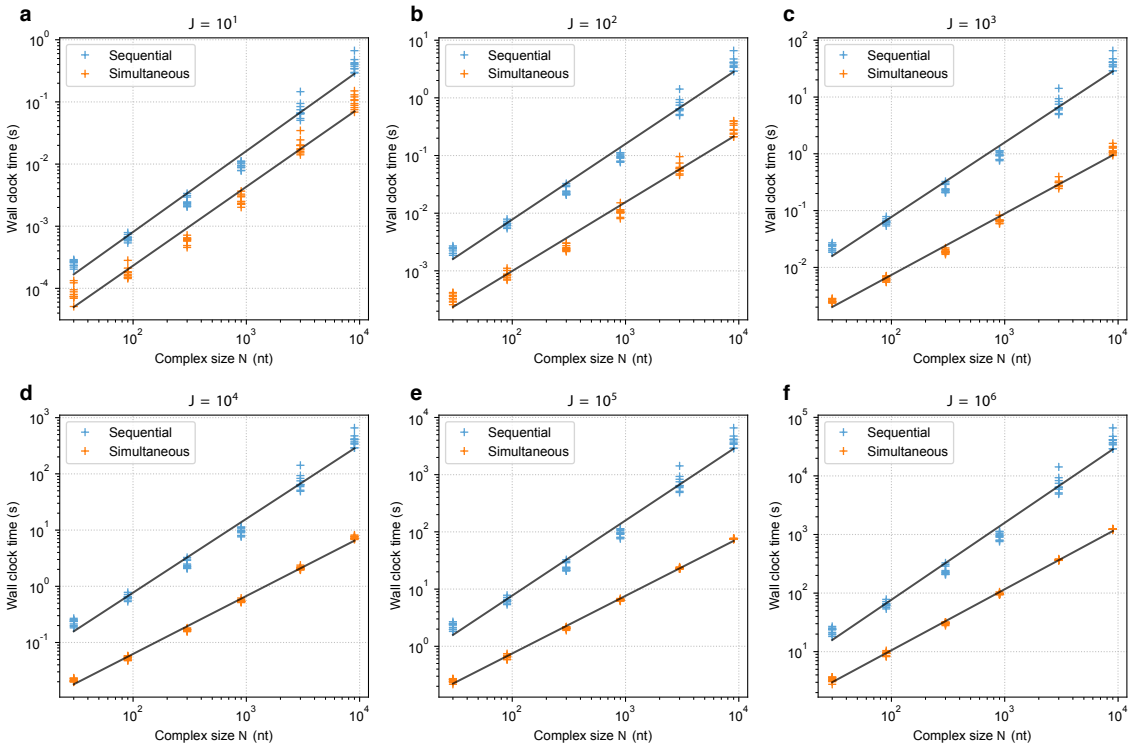


Figure S45: Cost of simultaneous and sequential structure sampling for designed complexes for different numbers of samples $J \in 10^1, \ldots, 10^6$. J structures are sampled for each of 10 replicate sets of sequences per complex size. Each data point represents the sampling time for one replicate. Lines represent univariate regressions (see Section S6.6.3)

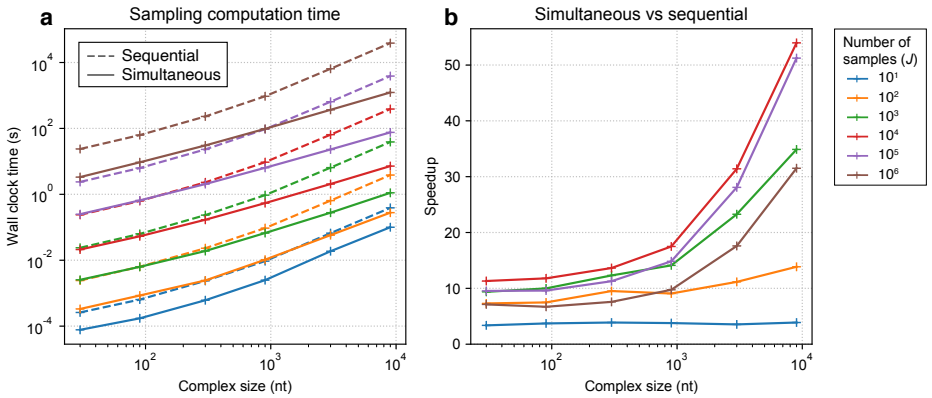


Figure S46: Sampling cost as a function of complex size (N) for designed complexes. (a) Cost of simultaneous and sequential sampling. Each data point represents the mean over all replicates for a given complex size. (b) Speedup using simultaneous vs sequential sampling. Each data point represents the ratio of means.

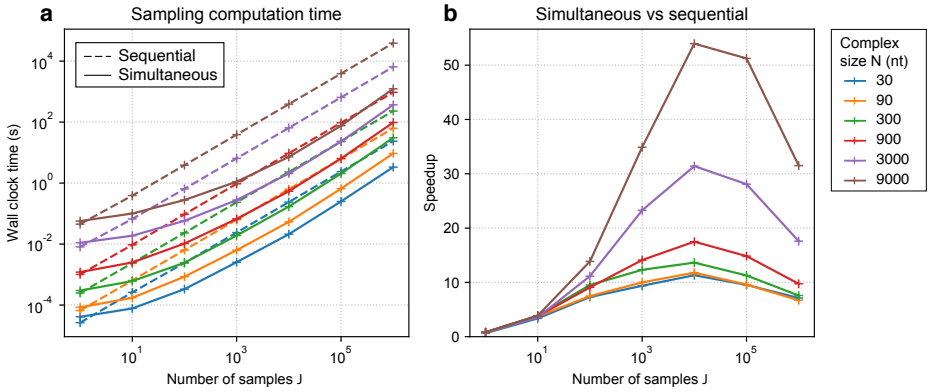


Figure S47: Sampling cost as a function of number of samples (J) for designed complexes. (a) Cost of simultaneous and sequential sampling. Each data point represents the mean over all replicates for a given complex size. (b) Speedup using simultaneous vs sequential sampling. Each data point represents the ratio of means. Not that for sufficiently large J, the cost of sorting becomes significant, lessening the speedup of the simultaneous approach.

S6.6.3 Empirical complexity estimates

Here, we measure empirical complexities for the sequential and simultaneous Boltzmann sampling algorithms using the random and designed complexes from Sections S6.6.1 and S6.6.2. Table A.3 uses bivariate least-squares linear regression to estimate the complexities with respect to N and J. For sequential sampling, the empirical complexity is $\sim J^{1.0}N^{1.3}$ for both random and designed complexes (the complexity in J is close to 1 because the calculation is just a repetition of a single sample J times). For simultaneous sampling, the empirical complexity is $\sim J^{0.8}N^{1.2}$ for random complexes and $\sim J^{0.8}N^{1.1}$ for designed complexes. Tables A.4 and A.5 use univariate least squares linear regressions to estimate the complexity with respect to J for fixed N and the complexity with respect to N for fixed J.

Sequences	Method	α_N	α_J	P (s)	r-value
Random	Sequential	1.2893	0.9913	2.190e-07	0.9982
Random	Simultaneous	1.1876	0.7806	3.437e-07	0.9902
Designed	Sequential	1.3126	0.9946	1.934e-07	0.9979
Designed	Simultaneous	1.1323	0.8094	2.799e-07	0.9848

Table A.3: Bivariate least-squares linear regression of sampling complexity. The fit is parametrized as $\log T \approx \alpha_N \log N + \alpha_J \log J + \log P$ such that $T \approx PN^{\alpha_N}J^{\alpha_J}$, with N the number of nucleotides in the complex, J the number of samples, T the computation time in seconds, α_N the complexity in N, α_J the complexity in J, and P the prefactor. For sequential sampling, α_J is close to 1 because the calculation is simply a repetition of a single sample J times.

Complexes	Method	N	α_J	P (s)	r-value
Random	Sequential	30	0.9879	2.608e-05	0.9998
Random	Sequential	90	0.9974	6.706e-05	0.9999
Random	Sequential	300	0.9932	2.410e-04	0.9998
Random	Sequential	900	0.9977	9.204e-04	0.9993
Random	Sequential	3000	0.9818	7.578e-03	0.9994
Random	Sequential	9000	0.9899	3.816e-02	0.9996
Random	Simultaneous	30	0.7975	2.444e-05	0.9881
Random	Simultaneous	90	0.8253	4.854e-05	0.9925
Random	Simultaneous	300	0.8158	1.807e-04	0.9881
Random	Simultaneous	900	0.8105	6.520e-04	0.9914
Random	Simultaneous	3000	0.7316	7.078e-03	0.9900
Random	Simultaneous	9000	0.7031	3.734e-02	0.9916
Designed	Sequential	30	0.9899	2.522e-05	0.9996
Designed	Sequential	90	0.9973	6.533e-05	0.9998
Designed	Sequential	300	0.9961	2.549e-04	0.9994
Designed	Sequential	900	0.9969	9.643e-04	0.9996
Designed	Sequential	3000	0.9922	7.679e-03	0.9979
Designed	Sequential	9000	0.9949	4.098e-02	0.9988
Designed	Simultaneous	30	0.8374	1.456e-05	0.9860
Designed	Simultaneous	90	0.8586	3.029e-05	0.9879
Designed	Simultaneous	300	0.8524	1.023e-04	0.9854
Designed	Simultaneous	900	0.8306	4.288e-04	0.9858
Designed	Simultaneous	3000	0.7570	3.662e-03	0.9759
Designed	Simultaneous	9000	0.7207	1.914e-02	0.9737

Table A.4: Univariate least-squares linear regression of sampling complexity in J The fit is parametrized as $\log T \approx \alpha_J \log J + \log P$ for fixed values of N such that $T \approx PJ^{\alpha_J}$, with N the number of nucleotides in the complex, J the number of samples, T the computation time in seconds, α_J the complexity in J, and P the prefactor. For sequential sampling, α_J is close to 1 because the calculation is simply a repetition of a single sample J times.

Complexes	Method	J	α_N	P (s)	r-value
Random	Sequential	10^1	1.2878	2.104e-06	0.9921
Random	Sequential	10^2	1.2855	2.063e-05	0.9929
Random	Sequential	10^3	1.2856	2.047e-04	0.9930
Random	Sequential	10^4	1.2870	2.032e-03	0.9929
Random	Sequential	10^5	1.2872	2.029e-02	0.9930
Random	Sequential	10^6	1.2870	2.034e-01	0.9930
Random	Simultaneous	10^1	1.3117	9.975e-07	0.9848
Random	Simultaneous	10^2	1.2836	5.124e-06	0.9896
Random	Simultaneous	10^3	1.2288	3.453e-05	0.9931
Random	Simultaneous	10^4	1.1360	3.800e-04	0.9960
Random	Simultaneous	10^5	1.0526	5.910e-03	0.9983
Random	Simultaneous	10^6	1.0575	7.958e-02	0.9992
Designed	Sequential	10^1	1.3032	1.996e-06	0.9906
Designed	Sequential	10^2	1.3111	1.853e-05	0.9914
Designed	Sequential	10^3	1.3130	1.826e-04	0.9915
Designed	Sequential	10^4	1.3142	1.809e-03	0.9916
Designed	Sequential	10^5	1.3145	1.806e-02	0.9916
Designed	Sequential	10^6	1.3145	1.805e-01	0.9916
Designed	Simultaneous	10^1	1.2708	6.636e-07	0.9866
Designed	Simultaneous	10^2	1.1939	4.066e-06	0.9908
Designed	Simultaneous	10^3	1.0799	5.096e-05	0.9954
Designed	Simultaneous	10^4	1.0315	5.403e-04	0.9981
Designed	Simultaneous	10^5	1.0074	7.228e-03	0.9990
Designed	Simultaneous	10^6	1.0420	8.692e-02	0.9991

Table A.5: Univariate least-squares linear regression of sampling complexity in N. The fit is parametrized as $\log T \approx \alpha_N \log N + \log P$ for fixed values of J such that $T \approx PN^{\alpha_N}$, with N the number of nucleotides in the complex, J the number of samples, T the computation time in seconds, α_N the complexity in N, and P the prefactor.

S6.7 Comparison of deterministic vs random MFE proxy structure estimation

Here, we compare the empirical performance of the deterministic and random approaches for estimating the MFE secondary structure over a complex ensemble with coaxial and dangle stacking subensembles (see Section S5.7). For recursions with coaxial and dangle stacking, the deterministic approach of using of the MinSum and ArgMin evaluation algebras yields the deterministic MFE proxy structure, $s_{\text{MFE}'}$, that contains the MFE stacking state $s^{||}_{\text{MFE}}$ within its subensemble (this is not guaranteed to be the MFE secondary structure s_{MFE}). Alternatively, with the random approach, the random MFE proxy structure, s_{MFE^*}, is the lowest free energy structure encountered within a random sample of J structures from the complex ensemble.

Comparisons are made for RNA sequences designed for the "multistranded engineered test set" of Reference [25] comprising target structures randomly assembled from duplex and loop sizes representative of the nucleic acid nanotechnology literature. Five independent sequence designs with ensemble defect $\leq 1\%$ were designed for each of 30 target structure for complex sizes $N \in \{100, 200, 400, 800, 1600, 3200\}$. MFE proxy structures were calculated using deterministic and random approaches for each of the 150 structures per complex size (using $J = 10^5$ samples for the random approach)

For this test set, the two methods typically yield proxy structures with similar free energies for smaller structures but the deterministic approach yields proxy structures with lower free energies as the complex ensemble gets larger (Figure S48ab). The equilibrium probability of the MFE proxy structure drops as the complex size increases (Figure S48cd), reducing the probability that the random approach discovers the true MFE for a fixed number of samples. Nonetheless, the MFE proxy structures generated by the two methods are structurally similar, having a median normalized base-pairing distance (defined below) that increases with complex size up to $\approx$1.5% for complexes with 3200 nt (Figure S48e). Note that the free energy of the MFE proxy structure, $\Delta G(\phi, s_{\text{MFE}'})$, is typically substantially lower than the free energy of the MFE stacking state, $\Delta G(\phi, s^{||}_{\text{MFE}})$, indicating that the MFE stacking state does not typically dominate the other stacking states in the subensemble of the secondary structure to which it contributes (Figure S48f).

The *normalized base-pairing distance* between two secondary structures, s_1 and s_2, containing N nucleotides each is the fraction of nucleotides paired differently in the two structures [25]:

$$d(s_1, s_2) = 1 - \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{N} S_{i,j}(s_1) S_{i,j}(s_2). \tag{S125}$$

Here, $S(s)$ is the *structure matrix* with entries defined as follows:

$$S_{i,j}(s) \equiv \begin{cases} 1 & i \neq j \text{ and structure } s \text{ contains base pair } i \cdot j \\ 1 & i = j \text{ and base } i \text{ is unpaired in structure } s \\ 0 & \text{otherwise.} \end{cases} \tag{S126}$$

Hence $S(s)$ is symmetric, the row sums of the augmented $S(s)$ matrix are unity, and $0 \leq d(s_1, s_2) \leq 1$.

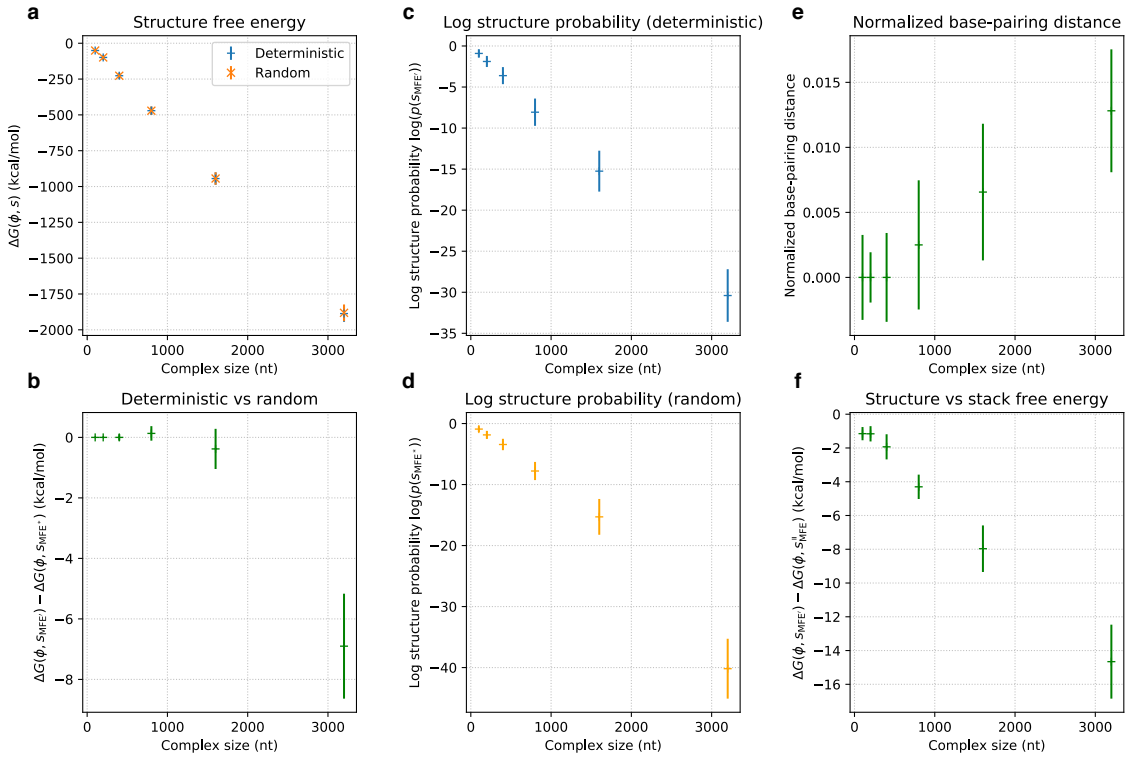


Figure S48: Comparison of deterministic vs random MFE proxy structure estimation. Each data point represents median ± median absolute deviation for 150 sequences per complex size. (a) Free energies of MFE proxy structures: deterministic ($\Delta G(\phi, s_{\text{MFE}'})$) and random ($\Delta G(\phi, s_{\text{MFE}^*}$). (b) Residuals from panel (a). (c) Equilibrium probability of the deterministic MFE proxy structure. (d) Equilibrium probability of the random MFE proxy structure. (e) Normalized base-pairing distance (S125) between the deterministic and random MFE proxy structures. (f) Comparison of structure free energy $\Delta G(\phi, s_{\text{MFE}'})$ and stacking state free energy $\Delta G(\phi, s''_{\text{MFE}})$ for deterministic MFE proxy structures.

S7 Validation

Here, we summarize the unit tests (Section S7.1) and regression tests (Section S7.2) used to validate the unified dynamic programming framework. Tests involving comparisons to enumerated quantities use the exhaustive enumeration algorithms described in Section S7.3.

S7.1 Unit tests

Over 100 C++ and Python unit tests were run via continuous integration on a dedicated JetBrains TeamCity server, comprising $O(10^7)$ test case assertions in total. Unless otherwise noted, unit tests for RNA employed `rna95` parameters and unit tests for DNA employed `dna04` parameters.

- **Individual loop free energies.** Test example loop free energies for all loop types vs manual calculations, including contributions from dangles and coaxial stacking, for each current parameter set (`rna95`, `rna06`, `dna04`).
- **Secondary structure enumeration.** Check structure counts, partition functions, and MFEs using $O(N^3)$ algorithms vs enumerated calculations, with wobble pairs on and off, for single and multiple random strands of DNA and RNA, with the recursions corresponding to any of the complex ensembles (`stacking`, `nostacking`, `none-nupack3`, `some-nupack3`, `all-nupack3`).
- **Partition functions and counts with coaxial and dangle stacking.** Check that $O(N^3)$ algorithms for the `stacking` complex ensemble match vectorized and unvectorized reference Python implementations of pseudocode for partition function and count for single and multiple random strands of DNA and RNA.
- **Overflow-safe evaluation algebra.** Check that overflow-safe partition function agrees with non-overflow variants for each complex ensemble (`stacking`, `nostacking`, `none-nupack3`, `some-nupack3`, `all-nupack3`), for single and multiple random strands of DNA and RNA.
- **Consistency between data types.** Verify that all results are equal for partition function calculations on complexes of up to 4 random RNA strands, using both $O(N^4)$ and $O(N^3)$ algorithms, for the following algorithms that transition between floating point formats to achieve overflow-safe performance: (1) 32 bit $\rightarrow$ 32 bit overflow-safe, (2) 32 bit $\rightarrow$ 64 bit overflow-safe, (3) 64 bit $\rightarrow$ 32 bit overflow-safe, (4) 32 bit $\rightarrow$ 64 bit non-overflow-safe $\rightarrow$ 32 bit overflow-safe. Perform this test for

each complex ensemble (`stacking`, `nostacking`, `none-nupack3`, `some-nupack3`, `all-nupack3`).

- **Consistency when using caching methodology for multistranded calculations.** Verify consistency of block caching used in multistranded algorithm for pair probability and partition function for random RNA strands, $O(N^3)$ and $O(N^4)$ algorithms, caching on and off, different orders of evaluations of requested complexes, on random sequences and edge cases we found during development. Perform this test for each complex ensemble (`stacking`, `nostacking`, `none-nupack3`, `some-nupack3`, `all-nupack3`).

- **Boltzmann sampled structure generation.** Estimate equilibrium structure probabilities and equilibrium base-pairing probability matrix from Boltzmann-sampled structures and check convergence to exact values as the number of samples increases. Perform this test for each complex ensemble (`stacking`, `nostacking`, `none-nupack3`, `some-nupack3`, `all-nupack3`) on single and multiple random strands of RNA.

- **Comparisons of different complexity algorithms.** Check that $O(N^3)$ and $O(N^4)$ algorithms agree for structure counts and partition functions. Perform this test for each complex ensemble (`stacking`, `nostacking`, `none-nupack3`, `some-nupack3`, `all-nupack3`) on single and multiple random strands of RNA.

S7.2 Regression tests

Regression tests were performed by comparing to results calculated using NUPACK 3.2.2 for historical complex ensembles and parameters sets supported by NUPACK 3.

- **Individual secondary structure free energies.** Test structure free energies vs NUPACK 3 for single and multiple random strands, wobble pairs on and off, random temperatures, historical complex ensembles (`none-nupack3`, `some-nupack3`, `all-nupack3`), and historical parameter sets (`rna95-nupack3`, `dna04-nupack3`, `rna99-nupack3`).

- **Necklace generation.** Test necklace generation for rotationally distinct strand orderings up to ($|\Psi_0| = 10$ strand species, $L_{\max} = 4$ strands per complex) vs NUPACK 3. Check that the number of free energies returned via dynamic programs is equal to the number of necklaces requested.

- **Partition functions and counts compared to NUPACK 3.** Check that structure counts and partition functions agree with NUPACK 3, for historical complex ensembles (`none-nupack3`, `some-nupack3`, `all-nupack3`), for historical parameter sets (`rna95-nupack3`, `dna04-nupack3`, `rna99-nupack3`), for wobble pairs on and off, for single and multiple random strands of RNA and DNA.

- **Comparison with NUPACK 3 for different parameter sets.** Check that structure counts and partition functions agree with NUPACK 3 for single and multiple random strands, for random temperatures, random concentrations of Na^+ (RNA or DNA) and Mg^{++} (DNA only), historical complex ensembles (`none-nupack3`, `some-nupack3`, `all-nupack3`), and historical parameter sets (`rna95-nupack3`, `dna04-nupack3`, `rna99-nupack3`).

- **MFE structures.** Check agreement with NUPACK 3 for single and multiple random strands of RNA and DNA, for historical ensembles (`none-nupack3`, `some-nupack3`, `all-nupack3`), for historical parameter sets (`rna95-nupack3`, `dna04-nupack3`, `rna99-nupack3`), for wobble pairs on and off.

- **Equilibrium base-pairing probability matrices.** Check matrices vs NUPACK 3 for single and multiple random strands of RNA and DNA, for historical complex ensembles (`none-nupack3`, `some-nupack3`, `all-nupack3`), for historical parameter sets (`rna95-nupack3`, `dna04-nupack3`, `rna99-nupack3`), for wobble pairs on and off. Test additional historical edge case sequences.

- **Suboptimal and MFE structures.** Check that generated structures for single and multiple random strands of DNA and RNA are identical to NUPACK 3 for 0 and 0.4 kcal/mol energy gaps, for historical complex ensembles (`none-nupack3`, `some-nupack3`, `all-nupack3`), for historical parameter sets (`rna95-nupack3`, `dna04-nupack3`, `rna99-nupack3`), for wobble pairs on and off. Test additional historical edge case sequences.

- **Equilibrium concentrations.** Check convergence and solution accuracy vs NUPACK 3 concentration solver using partition functions of random RNA complexes, random DNA complexes, and isolated edge cases that were found not to converge well in earlier versions of the code.

S7.3 Exhaustive enumeration algorithms

Here, we provide pseudocode for exhaustive enumeration algorithms that are used to help validate $O(N^4)$ and $O(N^3)$ dynamic programs on complexes that are small enough to permit exhaustive enumeration. Exhaustive enumeration is performed for a complex ensemble without coaxial and dangle stacking (Section S7.3.1), for enumeration of the coaxial and dangle stacking subensemble for a single secondary structure (Section S7.3.2), and for a complex ensemble with coaxial and dangle stacking (Section S7.3.3).

S7.3.1 Enumeration of complex ensemble without coaxial and dangle stacking subensembles

This pseudocode enumerates all possible secondary structures for a given complex ensemble (strand ordering) in a recursive manner. The implementation is chosen for its simplicity (rather than its efficiency), and relies on imposing a total ordering on base pair indices (i, j) via the function CompareBasePair. EnumerateSecondaryStructures is a generator function that yields all possible secondary structures by delegating to the inner generator function EnumerateHigherStructures. EnumerateSecondaryStructures yields all possible secondary structures including potentially disconnected structures that are not part of the complex ensemble. Any disconnected structures are removed in post-processing.

Algorithm S9: Enumeration of all secondary structures consistent with sequence ϕ.

```
CompareBasePair(p, p′)
1  i, j ← p
2  i′, j′ ← p′
3  return i < i′ or (i = i′ and j < j′)

EnumerateSecondaryStructures(φ)
1  N ← Length(φ)
2  s ← UnpairedStructure(N)
3  p ← (0, 0)
4  EnumerateHigherStructures(φ, s, p)

EnumerateHigherStructures(φ, s, p)
1  N ← Length(φ)
2  for i ∈ [1 : N]
3      for j ∈ [i + 1 : N]
4          p′ ← (i, j)
5          if CanPair(φ, i, j) and CompareBasePair(p, p′)
6              s′ ← AddBasePair(s, p′)
7              EnumerateHigherStructures(φ, s′, p′)
8  Yield s
```

S7.3.2 Enumeration of coaxial and dangle stacking subensemble for a single secondary structure

Stacking states are constructed hierarchically from a given secondary structure by first finding all coaxial stacking states (without dangles) for each loop and then finding all dangle stacking states consistent with each coaxial stacking state. The top-level function EnumerateStackingStatesForStructure is a generator function that yields all possible stacking states for a given secondary structure and sequence. EnumerateStackingStates-ForStructure does this by delegating to the function EnumerateLoopStackingStates, that yields all stacking states for a given loop within the secondary structure. EnumerateLoopStackingStates relies on the function GetValidMasks, that returns a list of all possible coaxial stacking states within the loop (neglecting dangles) and the function GetLoopStackingStates, that yields all stacking states for a loop consistent with a given coaxial stacking state.

The function Product takes a list of generators (or lists), G, and returns a tuple of elements, one from each generator (or list). This approach lazily generates all tuples from the cartesian product of the sets generated by each generator in G. (equivalent to a nested "for" loop with $|G|$ levels of nesting).

The function Type returns the type of loop ("hairpin," "stack," "bulge," "interior," "multi" or "exterior"). The function Subsequences splits the sequences of the loop into subsequences between the base pairs and nicks. For example, in a multiloop with base pairs $i \cdot j$, $d \cdot e$, and $f \cdot g$ with $i < d < e < f < g < j$, the function returns the list of subsequences $[\phi_{[i:d]}, \phi_{[e:f]}, \phi_{[g:j]}]$. In an exterior loop with a on the 3′ side of the nick and b on the 5′ side of the nick and base pairs $i \cdot j$ and $d \cdot e$ with $a < i < j < d < e < b$, the function returns the list of subsequences $[\phi_{[a:i]}, \phi_{[j:d]}, \phi_{[e:b]}]$. For each region of the loop, its stacking state is indicated with a number: 0 for no nucleotides stacking, 1 for a coaxial stack between the two adjacent base pairs, 3 for the 3′-most nucleotide stacking on the 3′ base pair, 5 for the 5′-most nucleotide stacking on the 5′ base pair, and 8 for the 3′-most nucleotide stacking on the 3′ base pair and the 5′-most nucleotide stacking on the 5′ base pair (if these are distinct nucleotides).

For a given subsequence, the function Before returns true if: (1) a base pair 5′-adjacent to the given subsequence is involved in a coaxial stack, or (2) there is a nick 5′-adjacent to the given subsequence. For a given subsequence, the function After returns true if: (1) a base pair 3′-adjacent to the given subsequence is involved in a coaxial stack, or (2) there is a nick 3′-adjacent to the given subsequence. These properties: (1) prevent nucleotides from

dangle stacking on a base pair that is already in a coaxial stack, and (2) prevent nucleotides adjacent to a nick from being included in invalid dangle states.

The function CoaxAdjacent returns true if a base pair either 5′- or 3′-adjacent to the given subsequence is involved in a coaxial stack. The function NickAdjacent returns true if there is a nick either 5′- or 3′-adjacent to the given subsequence.

Algorithm S10: Enumeration of all stacking states for a given sequence ϕ and secondary structure s.

EnumerateStackingStatesForStructure(ϕ, s)

```
G ← [ ]
for l ∈ Loops(s)
    Append(G, EnumerateLoopStackingStates(φ, l))
for [ω] ∈ Product(G) ▷ [ω] is a list of stacking states within each loop in s
    s^|| ← StackingState(s, [ω])
    Yield(s^||)
```

Algorithm S11: Enumeration of all stacking states for a given sequence ϕ and loop l. If the loop is not an exterior loop or multiloop, the function NoStacking returns an object indicating that the loop does not having a subensemble of stacking states (since coaxial and dangle stacking are defined only for exterior loops and multiloops).

EnumerateLoopStackingStates(ϕ, l)

```
φ^R ← Subsequences(φ, l)
V^mask ← GetValidMasks(φ^R, l)
if V^mask = [ ]
    Yield NoStacking()
for v^mask ∈ V^mask
    GetLoopStackingStates(φ^R, l, v^mask)
```

Algorithm S12: Enumeration of all coaxial stacking states in a given loop l containing sequence regions ϕ^R. The function BinaryVector(number, width) produces a vector of 1s and 0s that is the binary representation of the input number with zero-padding up to the input width.

GetValidMasks(ϕ^R, l)

if $\neg$(Type(l) = "multi" **or** Type(l) = "exterior")
    return [ ] ▷ No stacking states have to be enumerated
$v^{\text{indices}} \leftarrow$ [ ]
for $i \in [1 : |\phi^R|]$
    if $|\phi_i^R| = 2$
        Append(v^{indices}, i)

▷ Enumerate all possible combinations of coaxial stacks between base pairs in the loop.
$V^{\text{mask}} \leftarrow$ [ ]
for $i \in [0 : 2^{|v^{\text{indices}}|}]$
    ▷ Consider each combination of a base pair being in a coaxial stack or not.
    $t^{\text{mask}} \leftarrow$ BinaryVector($i, |v^{\text{indices}}|$)
    $v^{\text{mask}} \leftarrow$ [ ]
    for $i \in [1 : |\phi^R|]$
        if $i \in v^{\text{indices}}$
            Append($v^{\text{mask}}, t_i^{\text{mask}}$)
        else
            Append($v^{\text{mask}}, 0$)

    ▷ Filter out any mask that is invalid.
    $c \leftarrow$ true
    for $i \in [1 : |\phi^R|]$
        if $v_i^{\text{mask}} = 1$ **and** CoaxAdjacent($\phi_i^R, v^{\text{mask}}, l$)
            $c \leftarrow$ false
    if c = true
        Append($V^{\text{mask}}, v^{\text{mask}}$)
return V^{mask}

Algorithm S13: Enumeration of stacking states for given loop l containing sequence regions ϕ^R consistent with a given coaxial stacking state v^{mask}. LoopStackingState constructs a representation of a given loop l with a given list of stacks $[x]$.

GetLoopStackingStates($\phi^R, l, v^{\text{mask}}$)

```
1  G^R ← [ ]
2  ▷ Consider each subsequence region bounded by base pairs within the loop:
3  for i ∈ |φ^R|
4      if |φ_i^R| ≥ 3 or (|φ_i^R| ≥ 2 and NickAdjacent(φ_i^R))
5          if After(φ_i^R, v^mask, l) and ¬Before(φ_i^R, v^mask, l)
6              Append(G^R, [0, 5]) ▷ No dangles, or 5′ dangle
7
8          elseif Before(φ_i^R, v^mask, l) and ¬After(φ_i^R, v^mask, l)
9              Append(G^R, [0, 3]) ▷ No dangles, or 3′ dangle
10
11         elseif ¬(Before(φ_i^R, v^mask, l) or After(φ_i^R, v^mask, l))
12             if |φ_i^R| = 3
13                 Append(G^R, [0, 3, 5]) ▷ No dangles, 3′ dangle, or 5′ dangle
14             elseif |φ_i^R| > 3
15                 Append(G^R, [0, 3, 5, 8]) ▷ No dangles, 3′ dangle, 5′ dangle,
16                                           ▷ or both 3′ and 5′ dangles
17     else
18         Append(G^R, [v_i^mask])
19
20 ▷ Yield all possible combinations of dangle states.
21 for [x] ∈ Product(G^R)
22     ω ← LoopStackingState(l, [x])
23     Yield(ω)
```

S7.3.3 Enumeration of complex ensemble with coaxial and dangle stacking subensembles

To obtain all the stacking states for the complex, the above functions EnumerateSecondaryStructures and EnumerateStackingStatesForStructure are composed.

Algorithm S14: Enumeration of all stacking states consistent with sequence ϕ.

EnumerateStackingStates(ϕ)

1 **for** $s \in$ EnumerateSecondaryStructures(ϕ)

2 EnumerateStackingStatesForStructure(ϕ, s)

BIBLIOGRAPHY

[1] Mirela Andronescu, Anne Condon, Holger H. Hoos, David H. Mathews, and Kevin P. Murphy. Efficient parameter estimation for RNA secondary structure prediction. *Bioinformatics*, 23(13):i19–i28, 2007.

[2] Salvatore Bommarito, Nicolas Peyret, and John SantaLucia. Thermodynamic parameters for DNA sequences with dangling ends. *Nucleic Acids Research*, 28(9): 1929–1934, 2000.

[3] Ye Ding and Charles E. Lawrence. A statistical sampling algorithm for RNA secondary structure prediction. *Nucleic Acids Research*, 31(24):7280–7301, 2003.

[4] Robert M. Dirks and Niles A. Pierce. A partition function algorithm for nucleic acid secondary structure including pseudoknots. *Journal of Computational Chemistry*, 24 (13):1664–1677, 2003.

[5] Robert M. Dirks and Niles A. Pierce. An algorithm for computing nucleic acid base-pairing probabilities including pseudoknots. *Journal of Computational Chemistry*, 25 (10):1295–1304, 2004.

[6] Robert M. Dirks, Milo Lin, Erik Winfree, and Niles A. Pierce. Paradigms for computational nucleic acid design. *Nucleic Acids Research*, 32(4):1392–1403, 2004.

[7] Robert M. Dirks, Justin S. Bois, Joseph M. Schaeffer, Erik Winfree, and Niles A. Pierce. Thermodynamic analysis of interacting nucleic acid strands. *SIAM Review*, 49(1):65–88, 2007.

[8] Joseph A. Gallian. *Contemporary abstract algebra*. Chapman and Hall/CRC, 2021.

[9] Ryan T. Koehler and Nicolas Peyret. Thermodynamic properties of DNA sequences: Characteristic values for the human genome. *Bioinformatics*, 21(16):3333–3339, 2005.

[10] Zhi John Lu, Douglas H. Turner, and David H. Mathews. A set of nearest neighbor parameters for predicting the enthalpy change of RNA secondary structure formation. *Nucleic Acids Research*, 34(17):4912–4924, 2006.

[11] Rune B. Lyngsø, Michael Zuker, and C.N. Pedersen. Fast evaluation of internal loops in RNA secondary structure prediction. *Bioinformatics*, 15(6):440–445, 1999.

[12] David H. Mathews. How to benchmark RNA secondary structure prediction accuracy. *Methods*, 162:60–67, 2019.

[13] David H. Mathews and Douglas H. Turner. Experimentally derived nearest-neighbor parameters for the stability of RNA three-and four-way multibranch loops. *Biochemistry*, 41(3):869–880, 2002.

[14] David H. Mathews, Jeffrey Sabina, Michael Zuker, and Douglas H. Turner. Expanded sequence dependence of thermodynamic parameters improves prediction of RNA secondary structure. *Journal of Molecular Biology*, 288:911–940, 1999.

[15] 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.

[16] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[17] Nicolas Peyret. *Prediction of nucleic acid hybridization: Parameters and algorithms*. Thesis, Wayne State University, 2000.

[18] John SantaLucia. A unified view of polymer, dumbbell, and oligonucleotide DNA nearest-neighbor thermodynamics. *Proceedings of the National Academy of Sciences*, 95(4):1460–1465, 1998.

[19] John SantaLucia and Donald Hicks. The thermodynamics of DNA structural motifs. *Annual Review of Biophysics and Biomolecular Structure.*, 33:415–440, 2004.

[20] Martin J. Serra and Douglas H. Turner. Predicting thermodynamic properties of RNA. *Methods in Enzymology*, 259:242–261, 1995.

[21] Ignacio Tinoco, Olke C. Uhlenbeck, and Mark D. Levine. Estimation of secondary structure in ribonucleic acids. *Nature*, 230(5293):362–367, 1971.

[22] Douglas H. Turner and David H. Mathews. NNDB: The nearest neighbor parameter database for predicting stability of nucleic acid secondary structure. *Nucleic Acids Research*, 38(suppl_1):D280–D282, 2010.

[23] Brian R. Wolfe, Nicholas J. Porubsky, Joseph N. Zadeh, Robert M. Dirks, and Niles A. Pierce. Constrained multistate sequence design for nucleic acid reaction pathway engineering. *Journal of the American Chemical Society*, 139(8):3134–3144, 2017.

[24] Tianbing Xia, John SantaLucia, Mark E. Burkard, Ryszard Kierzek, Susan J. Schroeder, Xiaoqi Jiao, Christopher Cox, and Douglas H. Turner. Thermodynamic parameters for an expanded nearest-neighbor model for formation of RNA duplexes with Watson-Crick base pairs. *Biochemistry*, 37(42):14719–14735, 1998.

[25] Joseph N. Zadeh, Brian R. Wolfe, and Niles A. Pierce. Nucleic acid sequence design via efficient ensemble defect optimization. *Journal of Computational Chemistry*, 32 (3):439–452, 2011.

[26] Michael Zuker. Mfold web server for nucleic acid folding and hybridization prediction. *Nucleic Acids Research*, 31(13):3406–3415, 2003.