A Unified Dynamic Programming Framework for the Analysis of Interacting Nucleic Acid Strands: Enhanced Models, Scalability, and Speed

Mark E. Fornace,[†,#] **Nicholas J. Porubsky,**[†,#] **and Niles A. Pierce**[*,‡,§,¶]

[†]Division of Chemistry & Chemical Engineering, California Institute of Technology, Pasadena, CA 91125, USA. [‡]Division of Biology & Biological Engineering, California Institute of Technology, Pasadena, CA 91125, USA. [§]Division of Engineering & Applied Science, California Institute of Technology, Pasadena, CA 91125, USA. [¶]Weatherall Institute of Molecular Medicine, University of Oxford, Oxford OX3 9DS, UK.

ABSTRACT: 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 ≤ 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.

KEYWORDS: *RNA, DNA, base-pairing, secondary structure, equilibrium, complex ensemble, test tube ensemble, coaxial and dangle stacking subensembles, free energy model, dynamic programming algorithm, recursion, evaluation algebra, operation order, partition function, base-pairing probability, minimum free energy, concentration, structure sampling.*

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[1–13] to calculate diverse physical quantities.[10,14–22] We have previously developed dynamic programming algorithms that are unique in treating complex and test tube ensembles containing arbitrary numbers of interacting strand species,[22] 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.[23]

Here, following 15 years of methods development,[22–29] 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.

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.

PHYSICAL MODEL

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.

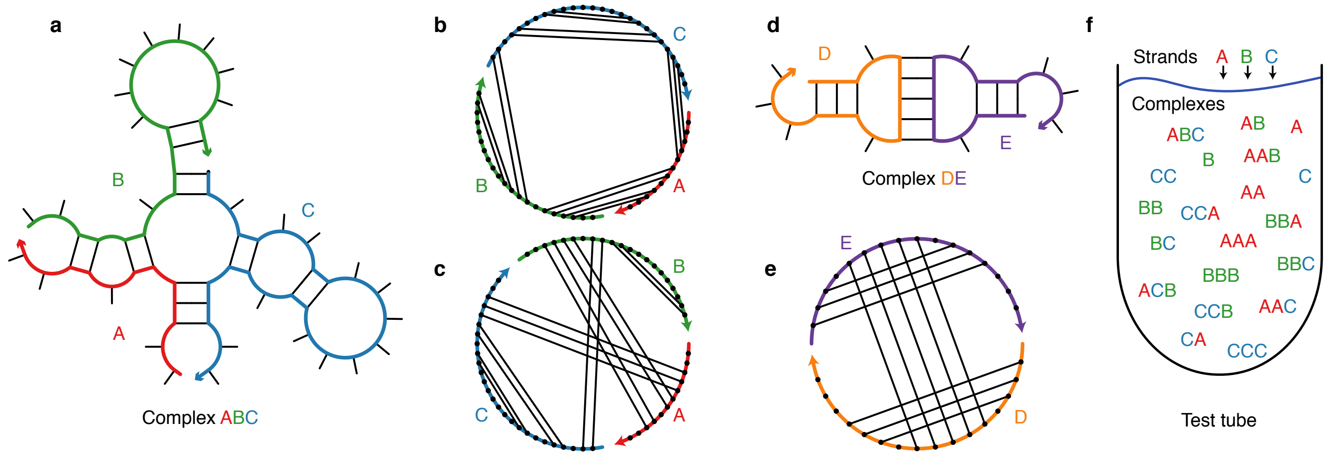


Figure 1: Complex and test tube ensembles. (a) A connected unpseudoknotted secondary structure for complex of 3 strands with strand ordering π = ABC. An arrowhead denotes the $3'$ end of each strand. (b) Polymer graph representation of the same secondary structure showing no crossing lines for strand 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.

- *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[30]) or large-scale structural engineering including pseudoknots (e.g., RNA origamis[31]), NUPACK generalizes sequence analysis and design to multi-complex and multi-tube ensembles.[29]

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 [A·U or C·G] or a wobble pair [G·U]. For DNA, the corresponding Watson–Crick pairs are A·T and C·G and there are no wobble pairs. Example secondary structures are displayed in Figures 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 1bc). A secondary structure is *unpseudoknotted* if there exists a strand ordering for which the polymer graph has no crossing lines (e.g., Figure 1b), or *pseudoknotted* if all strand orderings contain crossing lines (e.g., the kissing loops of Figure 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).[22] (We dispense with our prior convention[22,23,27] 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.[22]

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.,[22] 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 1f), although Ψ can be defined to contain arbitrary complex species formed from the strand species in Ψ^0.

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[3–5,8,12,13] plus a strand association penalty,[32] ΔG^{assoc}, applied $L-1$ times for a complex of L strands:

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

The secondary structure and polymer graph of Figure 2 illustrate the different loop types, with free energies modeled as follows:[3–5,8,12,13]

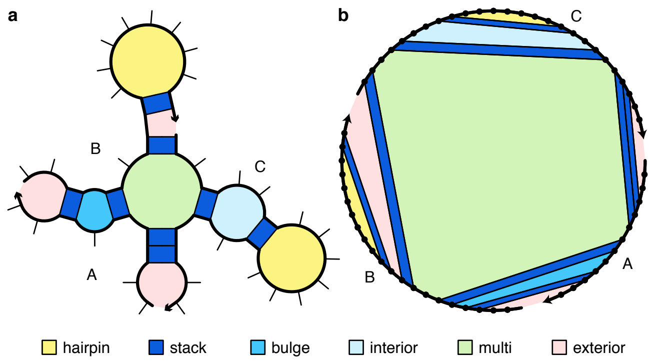


Figure 2: Loop-based free energy model for a complex. (a) Canonical loop types for complex with strand ordering $\pi =$ ABC. (b) Equivalent polymer graph representation. An arrowhead denotes the $3'$ end of each strand.

- 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. *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) $\Delta G^{\text{terminalbp}}_{i,j}$ 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 $\Delta G^{\text{terminalbp}}_{i,j}$ over all closing base pairs $i \cdot j$. Hence, an unpaired strand has a free energy of zero, corresponding to the reference state.[22]

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[10] by incorporating coaxial stacking[5,6,13] and dangle stacking[2,7,8,13] 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.[22] 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 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:

$$\Delta G^{\text{stacking}} = -kT \log \sum_{\omega \in \text{loop}} \prod_{x \in \omega} e^{-\Delta G_x / kT} \tag{2}$$

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 $s^{\shortparallel} \in s$ 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 $s^{\shortparallel} \in s$:

$$\overline{\Delta G}(\phi, s) = -kT \log \sum_{s^{\shortparallel} \in s} e^{-\overline{\Delta G}(\phi, s^{\shortparallel})/kT} \tag{3}$$

Let $\overline{\Gamma}^{\shortparallel}(\phi)$ denote the ensemble of stacking states corresponding to the complex ensemble of secondary structures $\overline{\Gamma}(\phi)$.

Symmetry Correction. 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 (1) must be adjusted[22] by a symmetry correction:

$$\Delta G(\phi, s) = \overline{\Delta G}(\phi, s) + \Delta G^{\text{sym}}(\phi, s). \tag{4}$$

where

$$\Delta G^{\text{sym}}(\phi, s) = kT \log R(\phi, s). \tag{5}$$

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 (1) and then the Distinguishability Correction Theorem of Dirks et al.[22] enables exact conversion of physical quantities to ensemble $\Gamma(\phi)$ using physical model (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).

Free Energy Parameters. Supported temperature-dependent RNA and DNA parameter sets include:

- `rna95` based on (Serra & Turner, 1995)[2] with additional parameters[8] including coaxial stacking[5,13] and dangle stacking[2,8,13] in 1M Na^+.
- `dna04` based on (SantaLucia, 1998)[3] and (SantaLucia & Hicks, 2004)[9] with additional parameters[8] including coaxial stacking[6] and dangle stacking[7,8] in user-specified concentrations of Na^+, K^+, NH_4^+, and Mg^{++} (see Section S1.2 for details on implementation of the salt corrections).[3,6,9,11]
- `rna06` based on (Mathews et al., 1999),[5] (Mathews et al., 2004),[10] and (Lu et al., 2006)[12] with additional parameters[4,8] including coaxial stacking[5,13] and dangle stacking[2,8,13] 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.

ALGORITHMS

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

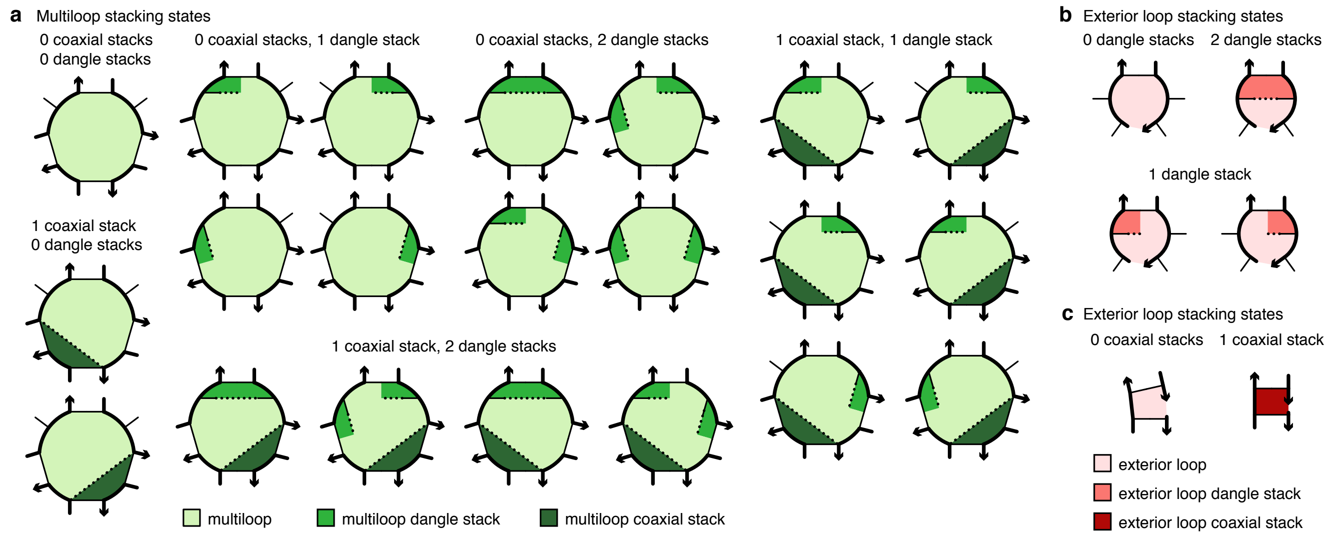

a Multiloop stacking states
0 coaxial stacks 0 dangle stacks
0 coaxial stacks, 1 dangle stack
0 coaxial stacks, 2 dangle stacks
1 coaxial stack, 1 dangle stack
1 coaxial stack 0 dangle stacks
1 coaxial stack, 2 dangle stacks
multiloop
multiloop dangle stack
multiloop coaxial stack
b Exterior loop stacking states
0 dangle stacks
2 dangle stacks
1 dangle stack
c Exterior loop stacking states
0 coaxial stacks
1 coaxial stack
exterior loop
exterior loop dangle stack
exterior loop coaxial stack

Figure 3: Coaxial and dangle stacking states for multiloops and exterior loops. (a) Stacking subensemble for the multiloop of Figure 2a. (b,c) Stacking subensembes for two exterior loops from Figure 2a.

- the partition function,

$$\overline{Q}(\phi) = \sum_{s\in\overline{\Gamma}(\phi)} e^{-\overline{\Delta G}(\phi,s)/kT}, \tag{6}$$

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

$$\overline{p}(\phi,s) = e^{-\overline{\Delta G}(\phi,s)/kT}/\overline{Q}(\phi). \tag{7}$$

Post-processing $\overline{Q}(\phi)$ yields the partition function $Q(\phi)$ over ensemble $\Gamma(\phi)$ treating strands with the same sequence as indistinguishable.[22]

- the base-pairing probability matrix $\overline{P}(\phi)$ with entries $\overline{P}^{i,j}(\phi) \in [0,1]$ corresponding to the probability

$$\overline{P}^{i,j}(\phi) = \sum_{s\in\overline{\Gamma}(\phi)} \overline{p}(\phi,s)S^{i,j}(s) \tag{8}$$

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 $s^{''}_{\rm MFE}(\phi) \in \overline{\Gamma}^{''}(\phi)$ treating all strands as distinct:

$$\overline{\Delta G}(\phi, s^{''}_{\rm MFE}) = \min_{s^{''}\in\overline{\Gamma}^{''}(\phi)} \overline{\Delta G}(\phi, s^{''}). \tag{9}$$

- the MFE proxy structure

$$s_{\rm MFE'} = \{s \in \overline{\Gamma}(\phi) | s^{''}_{\rm MFE} \in s, s^{''}_{\rm MFE}(\phi) = \arg\min_{s^{''}\in\overline{\Gamma}^{''}(\phi)} \overline{\Delta G}(\phi, s^{''})\}. \tag{10}$$

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.

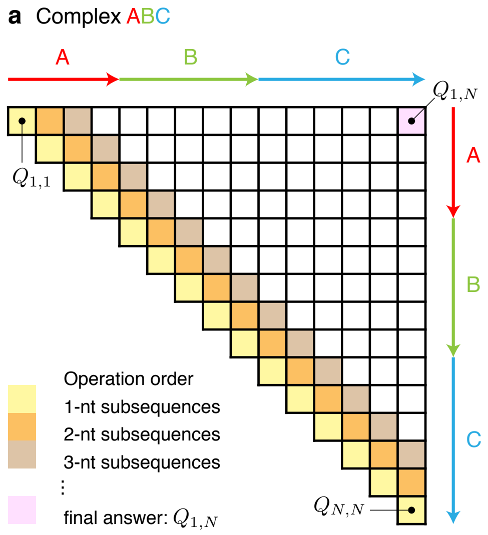


Figure 4: Operation order for partition function dynamic program over a complex ensemble with N nucleotides.

- the set of suboptimal secondary structures

$$\overline{\Gamma}_{\rm subopt}(\phi, \Delta G_{\rm gap}) = \{s \in \overline{\Gamma}(\phi) | s^{''} \in s, \overline{\Delta G}(\phi, s^{''}) \leq \overline{\Delta G}(\phi, s^{''}_{\rm MFE}) + \Delta G_{\rm gap}\} \tag{11}$$

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

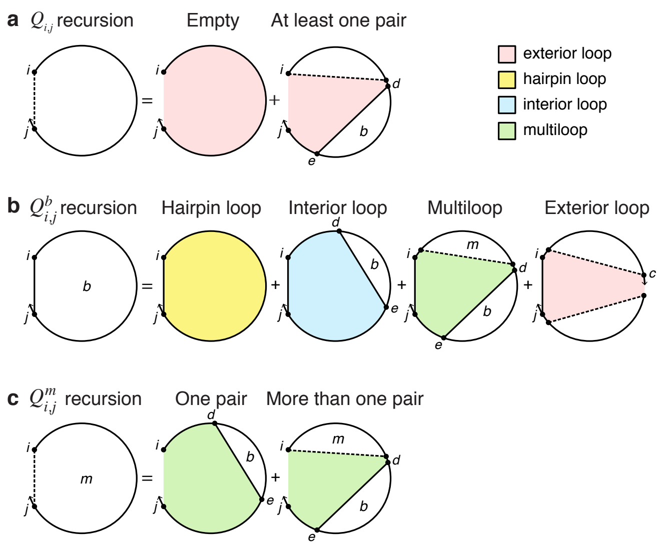
- a set of J secondary structures Boltzmann sampled from ensemble $\overline{\Gamma}(\phi)$ treating all strands as distinct:

$$\overline{\Gamma}_{\rm sample}(\phi, J) \in \overline{\Gamma}(\phi). \tag{12}$$

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

$$\Gamma_{\rm sample}(\phi, J) \in \Gamma(\phi). \tag{13}$$

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:



$$Q_{i,j} = 1 + \sum_{i \le d < e \le j} Q_{i,d-1}\, Q^b_{d,e}$$

$$\begin{aligned}
Q^b_{i,j} = {} & \exp\{-\Delta G^{\text{hairpin}}_{i,j}/kT\} + \sum_{i<d<e<j} Q^b_{d,e} \exp\{-\Delta G^{\text{interior}}_{i,d,e,j}/kT\} \\
& + \sum_{i<d<e<j} Q^m_{i+1,d-1} Q^b_{d,e} \exp\{-[\Delta G^{\text{multi}}_{\text{init}} + 2\Delta G^{\text{multi}}_{\text{bp}} + n_{e+1,j-1}\Delta G^{\text{multi}}_{\text{nt}}]/kT\} \\
& + \sum_{\substack{i \le c < j \\ \text{s.t. nick after } c}} Q_{i+1,c}\, Q_{c+1,j-1}
\end{aligned}$$

$$\begin{aligned}
Q^m_{i,j} = {} & \sum_{i \le d < e \le j} Q^b_{d,e} \exp\{-[\Delta G^{\text{multi}}_{\text{bp}} + (n_{i,d-1} + n_{e+1,j})\Delta G^{\text{multi}}_{\text{nt}}]/kT\} \\
& + \sum_{i \le d < e \le j} Q^m_{i,d-1}\, Q^b_{d,e} \exp\{-[\Delta G^{\text{multi}}_{\text{bp}} + n_{e+1,j}\Delta G^{\text{multi}}_{\text{nt}}]/kT\}
\end{aligned}$$

Figure 5: Partition function dynamic program recursion diagrams (left) and recursion equations (right).[22] 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). (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 (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).

- 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:[22]

$$\min_{x_\Psi} \sum_{c \in \Psi} x_c(\log x_c - \log Q_c - 1) \qquad (14a)$$

$$\text{subject to} \quad \sum_{c \in \Psi} A_{i,c} x_c = x^0_i \quad \forall i \in \Psi^0, \qquad (14b)$$

  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^0_i is the total concentration of strand i present in the test tube. Based on dimensional analysis,[22] the convex optimization algorithm operates on mole fractions, but for convenience, accepts molar strand concentrations $[i]^0 = x^0_i \rho_{H_2O}$ as inputs and returns molar complex concentrations $[c] = x_c \rho_{H_2O}$ as outputs, where ρ_{H_2O} is the molarity of water.
- the ensemble pair fractions for the test tube ensemble, for example

$$f_A(i_A \cdot j_B) \qquad (15)$$

  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 $\overline{P}_\Psi$.

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.[22] 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.[15,22] 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 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 5 left; with free energy contributions colored to match the loop types of Figure 2) or equivalently using recursion equations (Figure 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

Table 1: Algorithmic ingredients for calculating diverse physical quantities.

Quantity	Symbol	Recursions	Evaluation Algebra	Dependency	Operation Order
Partition function	$\overline{Q}(\phi)$	Stacking	SumProduct, SplitExp	–	Blockwise forward sweep
MFE	$\overline{\Delta G}(\phi, s^{\shortparallel}_{\rm MFE})$	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	$\overline{\Gamma}_{\rm sample}(\phi, J)$	Stacking	ArgRandJ	$\overline{Q}(\phi)$	Backtracking, priority queue
MFE structure proxy	$s_{\rm MFE'}(\phi)$	Stacking	ArgMin	$\overline{\Delta G}(\phi, s^{\shortparallel}_{\rm MFE})$	Backtracking, stack
Suboptimal ensemble	$\overline{\Gamma}_{\rm subopt}(\phi, \Delta G_{\rm gap})$	Stacking	ArgMinGap	$\overline{\Delta G}(\phi, s^{\shortparallel}_{\rm MFE})$	Backtracking, stack
Concentrations	x_Ψ	–	–	Q_Ψ	Convex optimization
Ensemble pair fractions	$f_A(i_A \cdot j_B)$	–	–	$x_\Psi, \overline{P}_\Psi$	–

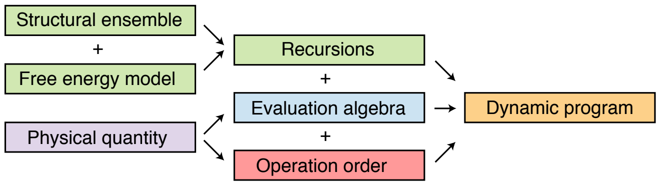


Figure 6: Unified dynamic programming framework. To calculate a physical quantity of interest based on a physical model comprising a structural ensemble and a free energy model, each dynamic program combines three ingredients: model-specific recursions, a quantity-specific evaluation algebra, and a quantity-specific operation order.

free energy model (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, $\overline{P}(\phi)$,[15,22,26] or to Boltzmann sample a structure from ensemble $\overline{\Gamma}(\phi)$.[18,22]

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, $\overline{\Delta G}(\phi, s_{\rm MFE}) = F_{1,N}$.[14,22] 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), $s_{\rm MFE}(\phi) \in \overline{\Gamma}(\phi)$, or the ensemble of suboptimal structures, $\overline{\Gamma}_{\rm subopt}(\phi, \Delta G^{\rm gap})$. 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.

Unified Dynamic Programming Framework. In the new unified framework, each dynamic program combines three ingredients (Figure 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 1).

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) and illustrated in Figure 3. For the recursions without coaxial and dangle stacking, the elementary recursion entity is a *terminal base pair* (Figure 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 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

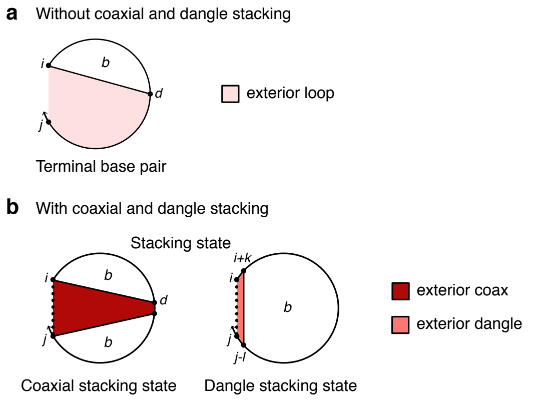


Figure 7: Elementary recursion entities without or with coaxial and dangle stacking. (a) Terminal base pair: a base pair that terminates a duplex $(i \cdot d)$ in an exterior loop or multiloop context. (b) Stacking state: (1) Either a coaxial stacking state: two adjacent terminal base pairs that are 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 j 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.

	Algebra	Algorithm Output	$\mathbb{0}$	$\mathbb{1}$	$a \oplus b$	$a \otimes b$	$W(g)$
a	SumProduct	Partition function	0	1	$a+b$	$a \cdot b$	$\exp(-g/kT)$
	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	$a_\mathrm{m} \cdot 2^{a_\mathrm{e}+\gamma} + b_\mathrm{m} \cdot 2^{b_\mathrm{e}+\gamma}$	$a_\mathrm{m} \cdot b_\mathrm{m}$	$\exp(-g/kT)$
	Exponent		0	γ	0	$a_\mathrm{e} + b_\mathrm{e} + \gamma$	γ
c	ArgRand	Sampled structure					
	Value		0	1	$a_\mathrm{v} + b_\mathrm{v}$	$a_\mathrm{v} \cdot b_\mathrm{v}$	$\exp(-g/kT)$
	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: 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 $S3$ for details.

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.

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 5a can equivalently be expressed as the partition function recursion equations of Figure 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 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 to produce a suite of executables for calculating the corresponding physical quantities.

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).[22]

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 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 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,

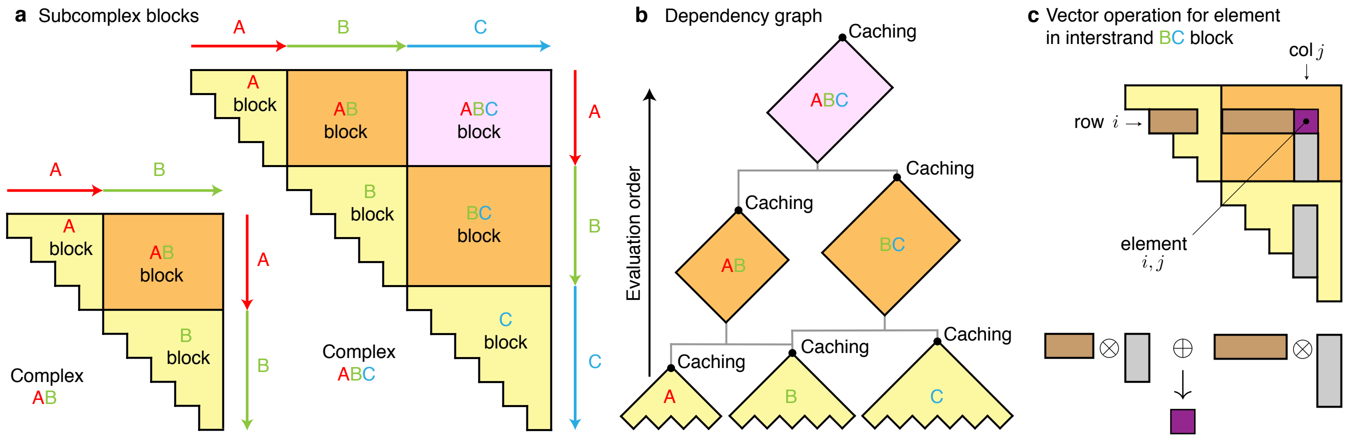


Figure 8: Blockwise operation order for dynamic programs operating on complex and test tube ensembles. (a) Subcomplex blocks within dynamic programming matrices (cf. Figure 4): triangular intrastrand blocks (A, B, C) and rectangular interstrand blocks (AB, BC, ABC) for complexes AB and ABC. Element i, 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.

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.

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.[22] 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 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.

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 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 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.[17,24] Moreover, with this blockwise operation order, recursions (Section S2) can be coded using vectorized dot products (Figure 8c) such that compilation with the appropriate evaluation algebra (Table 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 9.

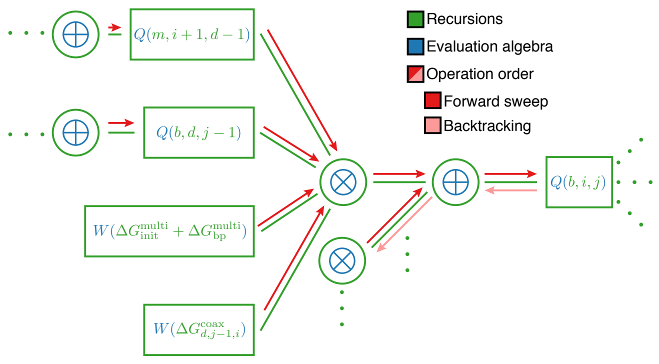

$Q(m, i+1, d-1)$
$Q(b, d, j-1)$
$W(\Delta G^{\text{multi}}_{\text{init}} + \Delta G^{\text{multi}}_{\text{bp}})$
$W(\Delta G^{\text{coax}}_{d,j-1,i})$
$Q(b, i, j)$
🟩 Recursions
🟦 Evaluation algebra
Operation order
🟥 Forward sweep
Backtracking

Figure 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.

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

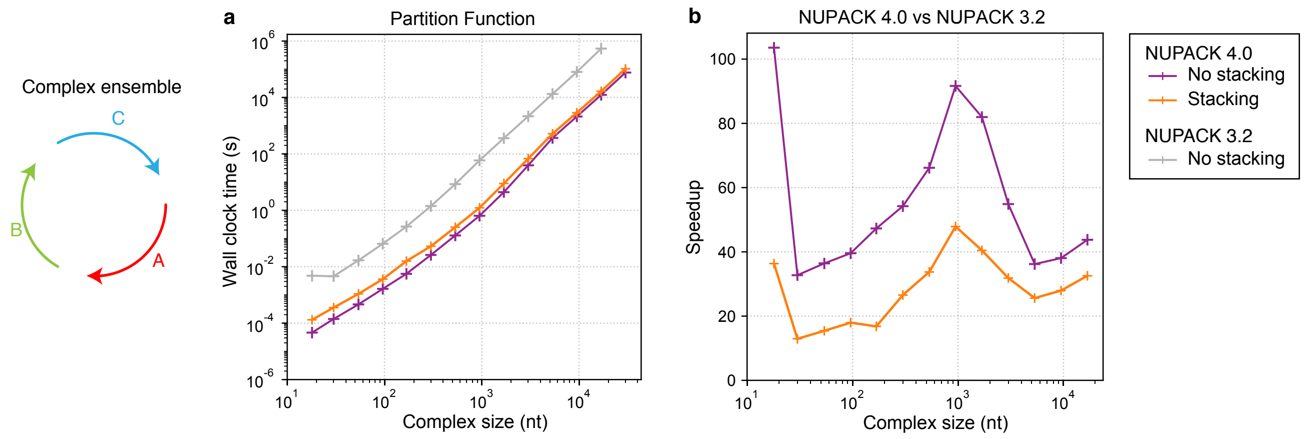


Figure 10: Enhanced efficiency for partition function calculations on complex ensembles including very large complexes. Calculation of the partition function for a complex of 3 strands, each with a different random sequence of uniform length. NUPACK 4.0 (vectorized, overflow-safe implementation, physical 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$^+$.

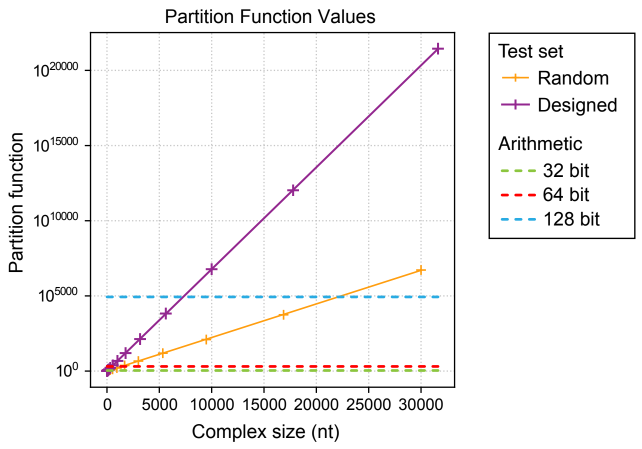


Figure 11: Overflow-safe partition function calculations on complex ensembles including very large complexes. Dashed lines denote the overflow thresholds for single-, double-, and quadruple-precision arithmetic. Partition function calculations performed using NUPACK 4.0 (overflow-safe implementation 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$^+$.

operating on the simpler physical model that neglects these terms. Figure 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.

Enhanced efficiency of the partition function algorithm for sets of complexes in test tube ensembles. Figure 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 complexes of up to $L_{\max}$ strands (Figure 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 $\approx$20× for test tubes containing all complexes of up to $L_{\max} = 2$ strands and up to $\approx$120× for test tubes containing all complexes up to $L_{\max} = 6$ strands. With NUPACK 4.0, Figure 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).

Backtrack-free base-pairing probability matrices. Historically, equilibrium base-pairing probabilities for a single strand[15,25] or a complex[22] are calculated using a dynamic program that backtracks through the matrix of subsequence partition functions. This backtracking process involves subtraction of intermediate 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 π = 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 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

$$\overline{P}_{i,j}(\phi) = Q^b_{i,j}(\phi) Q^{b_{\text{ext}}}_{i,j}(\phi) / Q_{1,N}(\phi). \tag{16}$$

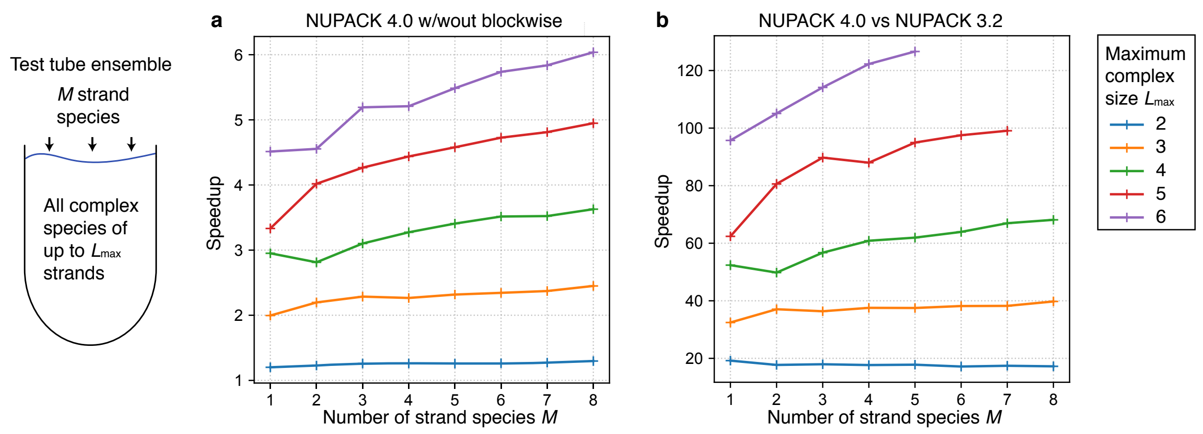


Figure 12: Enhanced efficiency of the partition function algorithm for sets of complexes in test tube ensembles. Calculation of the partition function for all complexes of up to $L_{\max}$ strands for a test tube ensemble containing M strand species, each with a different random 50 nt sequence. (a) Speedup 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.

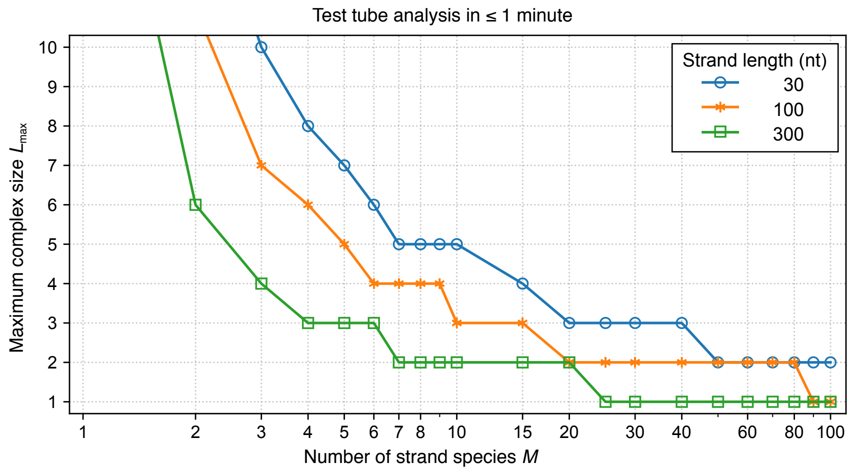


Figure 13: Equilibrium test tube analysis in under 1 minute. Calculation of the partition function and equilibrium complex concentration for a test tube ensemble containing M strand species that form all complexes of up to $L_{\max}$ strands. Symbols denote test tube ensembles for which the wall clock 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 (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.

Mathews employed this approach using new recursions to calculate the external conditional partition function $Q^{b_{\text{ext}}}_{i,j}$ for a single strand.[10] 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 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 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:

$$\overline{P}_{i,j}(\phi) = Q^b_{i,j}(\phi)Q^b_{j,N+i}(\phi')/Q_{1,N}(\phi). \qquad (17)$$

In Figure 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 ≈1.5–3C_Q (Figure S40).

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[18] or a complex,[22] 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[14] or a complex,[22] 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 6) using the same set of recursion diagrams/equations (Section S2) as the forward algorithms, but employing new evaluation algebras (Table 2cd), and with the operation order reversed so the blockwise dependency tree (Figure 8b) is traversed top to

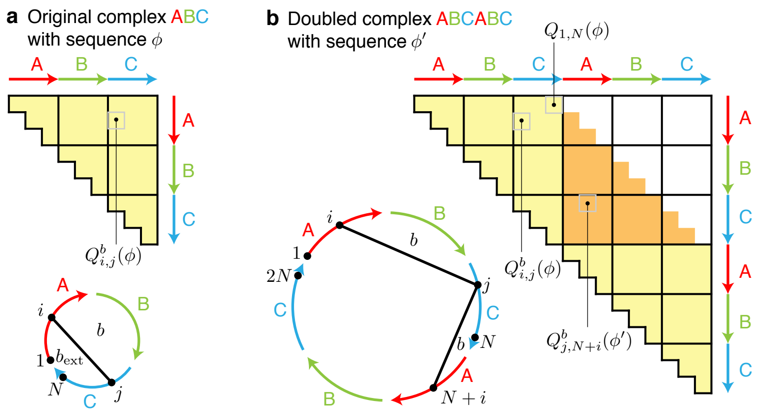


Figure 14: Backtrack-free calculation of the equilibrium base-pairing probability $P_{i,j}(\phi)$ for a complex ABC of N nucleotides with sequence ϕ using (17) and the conditional partition functions $Q^b_{i,j}(\phi)$ and $Q^b_{j,N+i}(\phi')$. The latter is calculated by considering the "doubled" complex ABCABC of $2N$ nucleotides with sequence ϕ'.

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 2cd). For sampling, $\oplus$ corresponds to randomly choosing between competing (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 $\Gamma(\phi, \Delta G_{\text{gap}})$ 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?"[18] or "is the free energy landscape dominated by multiple deep basins each defined by a set of related secondary structures?"[33] Existing algorithms perform sequential sampling of J structures for a strand[18] ($O(JN^2)$ time complexity if long interior loops are excluded) or a complex[22] ($O(JN^3)$ 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 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).

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[27–29] 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.

METHODS SUMMARY

Implementation. NUPACK algorithms are programmed in the C++17 programming language. Dynamic programs

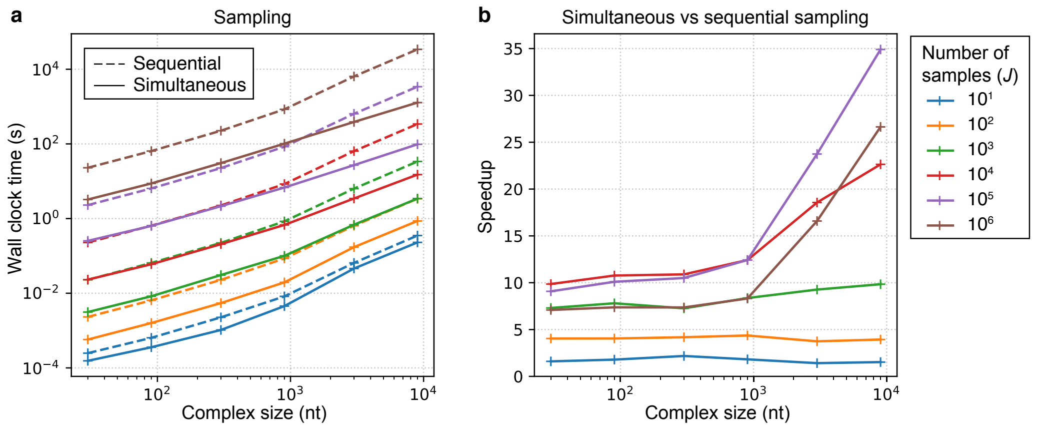


Figure 15: Enhanced efficiency for sampling multiple structures from complex ensembles using simultaneous rather than sequential sampling. Boltzmann sampling secondary structures for a complex of 3 strands, each with a different random sequence of uniform length. (a) Computational cost. (b) Computational 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.

are implemented using a generic programming paradigm[34] 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. Single-threaded single instruction multiple data (SIMD) vectorization is implemented using the Boost.SIMD library.[35] The convex optimization problem (14) is solved in the dual form using an efficient trust region method[22] using the Armadillo linear algebra library for matrix operations.[36]

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 10).

RESOURCES

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.

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.

Technical Support. Please direct questions, comments, feature requests, and bug reports to `support@nupack.org`.

ASSOCIATED CONTENT

Supporting Information

Additional free energy model details, recursions for the complex ensemble with or without coaxial and dangle stacking, evaluation algebras for each physical quantity, operation orders for each physical quantity, distinguishability issues, additional studies, validation.

AUTHOR INFORMATION

Corresponding Author

*E-mail: niles@caltech.edu

ORCID

Mark E. Fornace: 0000-0002-5829-5839
Nicholas J. Porubsky: 0000-0001-6330-2645
Niles A. Pierce: 0000-0003-2367-4406

Author Contributions

#M.E.F and N.J.P. contributed equally.

Notes

The authors declare no competing financial interests.

ACKNOWLEDGMENTS

We thank all the NUPACK users that have helped out as beta testers over the years, as well as the many NUPACK users that have emailed `support@nupack.org` to request features or report bugs. We thank J.S. Bois for helpful discussions, J. Huang for assistance developing the NUPACK Python module, and S.J. Schulte for performing preliminary studies. This work was funded by the National Science Foundation (Software Elements NSF-OAC-1835414, INSPIRE NSF-CHE-1643606, Molecular Programming Project NSF-CCF-1317694), by the Programmable Molecular Technology Center (PMTC) within the Beckman Institute at Caltech, by the AWS/IST Cloud Credit Program at Caltech, by a Microsoft Azure sponsorship, by the National Institutes of Health (National Research Service Award T32 GM007616), by a Professorial Fellowship at Balliol College, University of Oxford, and by the Eastman Visiting Professorship at the University of Oxford.

REFERENCES

(1) Tinoco, I., Jr., Uhlenbeck, O. C., and Levine, M. D. (1971) Estimation of secondary structure in ribonucleic acids. *Nature* 230, 362–367.

(2) Serra, M. J., and Turner, D. H. (1995) Predicting thermodynamic properties of RNA. *Methods Enzymol.* 259, 242–261.

(3) SantaLucia, J., Jr. (1998) A unified view of polymer, dumbbell, and oligonucleotide DNA nearest-neighbor thermodynamics. *Proc. Natl. Acad. Sci.* 95, 1460–1465.

(4) Xia, T. B., SantaLucia, J., Jr., Burkard, M. E., Kierzek, R., Schroeder, S. J., Jiao, X. Q., Cox, C., and Turner, D. H. (1998) Thermodynamic parameters for an expanded nearest-neighbor model for formation of RNA duplexes with Watson-Crick base pairs. *Biochemistry* 37, 14719–14735.

(5) Mathews, D. H., Sabina, J., Zuker, M., and Turner, D. H. (1999) Expanded sequence dependence of thermodynamic parameters improves prediction of RNA secondary structure. *J. Mol. Biol.* 288, 911–940.

(6) Peyret, Nicolas, *Prediction of nucleic acid hybridization: parameters and algorithms.* Thesis, Wayne State University (2000).

(7) Bommarito, S., Peyret, N., and SantaLucia, J. (2000) Thermodynamic parameters for DNA sequences with dangling ends. *Nucleic Acids Res.* 28, 1929–1934.

(8) Zuker, M. (2003) Mfold web server for nucleic acid folding and hybridization prediction. *Nucleic Acids Res.* 31, 3406–3415.

(9) SantaLucia, J., Jr., and Hicks, D. (2004) The thermodynamics of DNA structural motifs. *Annu. Rev. Biophys. Biomol. Struct.* 33, 415–440.

(10) Mathews, D. H., Disney, M. D., Childs, J. L., Schroeder, S. J., Zuker, M., and Turner, D. H. (2004) Incorporating chemical modification constraints into a dynamic programming algorithm for prediction of RNA secondary structure. *Proc. Natl. Acad. Sci.* 101, 7287–7292.

(11) Koehler, R. T., and Peyret, N. (2005) Thermodynamic properties of DNA sequences: characteristic values for the human genome. *Bioinformatics* 21, 3333–3339.

(12) Lu, Z. J., Turner, D. H., and Mathews, D. H. (2006) A set of nearest neighbor parameters for predicting the enthalpy change of RNA secondary structure formation. *Nucleic Acids Res.* 34, 4912–4924.

(13) Turner, D. H., and Mathews, D. H. (2010) NNDB: The nearest neighbor parameter database for predicting stability of nucleic acid secondary structure. *Nucleic Acids Res.* 38, D280–D282.

(14) Zuker, M., and Stiegler, P. (1981) Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. *Nucleic Acids Res.* 9, 133–147.

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

(16) Hofacker, I. L., Fontana, W., Stadler, P. F., Bonhoeffer, L. S., Tacker, M., and Schuster, P. (1994) Fast folding and comparison of RNA secondary structures. *Chem. Mon.* 125, 167–188.

(17) Lyngsø, R. B., Zuker, M., and Pedersen, C. N. S. (1999) Fast evaluation of internal loops in RNA secondary structure prediction. *Bioinformatics* 15, 440–445.

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

(19) Dimitrov, R. A., and Zuker, M. (2004) Prediction of hybridization and melting for double-stranded nucleic acids. *Biophys. J.* 87, 215–226.

(20) Andronescu, M., Zhang, Z. C., and Condon, A. (2005) Secondary structure prediction of interacting RNA molecules. *J. Mol. Biol.* 345, 987–1001.

(21) Bernhart, S. H., Tafer, H., Muckstein, U., Flamm, C., Stadler, P. F., and Hofacker, I. L. (2006) Partition function and base pairing probabilities of RNA heterodimers. *Algorithms Mol. Biol.* 1.

(22) Dirks, R. M., Bois, J. S., Schaeffer, J. M., Winfree, E., and Pierce, N. A. (2007) Thermodynamic analysis of interacting nucleic acid strands. *SIAM Rev.* 49, 65–88.

(23) Zadeh, J. N., Steenberg, C. D., Bois, J. S., Wolfe, B. R., Pierce, M. B., Khan, A. R., Dirks, R. M., and Pierce, N. A. (2011) NUPACK: Analysis and design of nucleic acid systems. *J. Comput. Chem.* 32, 170–173.

(24) Dirks, R. M., and Pierce, N. A. (2003) A partition function algorithm for nucleic acid secondary structure including pseudoknots. *J. Comput. Chem.* 24, 1664–1677.

(25) Dirks, R. M., Lin, M., Winfree, E., and Pierce, N. A. (2004) Paradigms for computational nucleic acid design. *Nucleic Acids Res.* 32, 1392–1403.

(26) Dirks, R. M., and Pierce, N. A. (2004) An algorithm for computing nucleic acid base-pairing probabilities including pseudoknots. *J. Comput. Chem.* 25, 1295–1304.

(27) Zadeh, J. N., Wolfe, B. R., and Pierce, N. A. (2011) Nucleic acid sequence design via efficient ensemble defect optimization. *J. Comput. Chem.* 32, 439–452.

(28) Wolfe, B. R., and Pierce, N. A. (2015) Nucleic acid sequence design for a test tube of interacting nucleic acid strands. *ACS Synth. Biol.* 4, 1086–1100.

(29) Wolfe, B. R., Porubsky, N. J., Zadeh, J. N., Dirks, R. M., and Pierce, N. A. (2017) Constrained multistate sequence design for nucleic acid reaction pathway engineering. *J. Am. Chem. Soc.* 139, 3134–3144.

(30) Hochrein, L. M., Schwarzkopf, M., Shahgholi, M., Yin, P., and Pierce, N. A. (2013) Conditional Dicer substrate formation via shape and sequence transduction with small conditional RNAs. *J. Am. Chem. Soc.* 135, 17322–17330.

(31) Geary, C., Rothemund, P. W. K., and Andersen, E. S. (2014) A single-stranded architecture for cotranscriptional folding of RNA nanostructures. *Science* 345, 799–804.

(32) Bloomfield, V. A., Crothers, D. M., and Tinoco, I., Jr. (2000) *Nucleic acids: structures, properties, and functions* (University Science Books, Sausalito, CA).

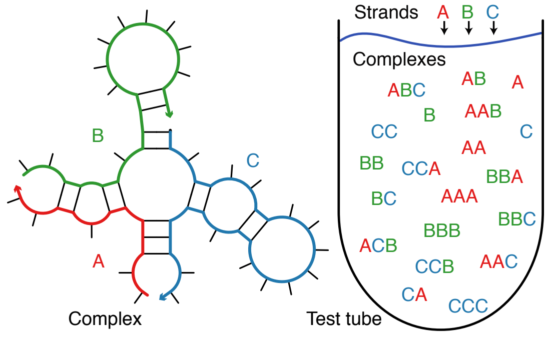
(33) Ding, Y., Chan, C. Y., and Lawrence, C. E. (2005) RNA secondary structure prediction by centroids in a Boltzmann weighted ensemble. *RNA* 11, 1157–1166.

(34) Stepanov, A. A., and Rose, D. E. (2014) *From mathematics to generic programming* (Pearson Education, Crawfordsville, Indiana).

(35) Esterie, P., Falcou, J., Gaunard, M., and Lapreste, J.-T. (2012) Boost.SIMD: Generic programming for portable SIMDization. In *Proceedings of the 2014 workshop on programming models for SIMD/vector processing* (ACM, New York). pp. 1–8.

(36) Sanderson, C., and Curtin, R. (2016) Armadillo: A template-based C++ library for linear algebra. *J. Open Source Softw.* 1, 26.

For Table of Contents Use Only



Title: A Unified Dynamic Programming Framework for the Analysis of Interacting Nucleic Acid Strands: Enhanced Models, Scalability, and Speed

Authors: Mark E. Fornace, Nicholas J. Porubsky, and Niles A. Pierce