Minimum Free Energy, Partition Function and Kinetics Simulation Algorithms for a Multistranded Scaffolded DNA Computer

Ahmed Shalaby
Hamilton Institute, Department of Computer Science, Maynooth University, Ireland

Chris Thachuk
Paul G. Allen School of Computer Science & Engineering, University of Washington, Seattle, WA, USA

Damien Woods
Hamilton Institute, Department of Computer Science, Maynooth University, Ireland

Abstract

Polynomial time dynamic programming algorithms play a crucial role in the design, analysis and engineering of nucleic acid systems including DNA computers and DNA/RNA nanostructures. However, in complex multistranded or pseudoknotted systems, computing the minimum free energy (MFE), and partition function of nucleic acid systems is NP-hard. Despite this, multistranded and/or pseudoknotted systems represent some of the most utilised and successful systems in the field. This leaves open the tempting possibility that many of the kinds of multistranded and/or pseudoknotted systems we wish to engineer actually fall into restricted classes, that do in fact have polynomial time algorithms, but we've just not found them yet.

Here, we give polynomial time algorithms for MFE and partition function calculation for a restricted kind of multistranded system called the 1D scaffolded DNA computer. This model of computation thermodynamically favours correct outputs over erroneous states, simulates finite state machines in 1D and Boolean circuits in 2D, and is amenable to DNA storage applications. In an effort to begin to ask the question of whether we can naturally compare the expressivity of nucleic acid systems based on the computational complexity of prediction of their preferred energetic states, we show our MFE problem is in logspace (the complexity class L), making it perhaps one of the simplest known, natural, nucleic acid MFE problems. Finally, we provide a stochastic kinetic simulator for the 1D scaffolded DNA computer and evaluate strategies for efficiently speeding up this thermodynamically favourable system in a constant-temperature kinetic regime.

2012 ACM Subject Classification Theory of computation → Models of computation; Applied computing → Physical sciences and engineering

Keywords and phrases thermodynamic computation, model of computation, molecular computing, minimum free energy, partition function, DNA computing, DNA self-assembly, DNA strand displacement, kinetics simulation

Digital Object Identifier 10.4230/LIPIcs.DNA.29.1

Supplementary Material *Software (Source code)*: `https://dna.hamilton.ie/software.html`

Funding Damien Woods and Ahmed Shalaby were supported by European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 772766, Active-DNA project), and Science Foundation Ireland (SFI) under grant numbers 20/FFP-P/8843 and 18/ERCS/5746. Chris Thachuk was supported by a US NSF grant (CCF 2211792) and a Faculty Early Career Development Award from NSF (CCF 2143227).

Acknowledgements We thank Abeer Eshra for extensive discussions on the thermodynamics and kinetics of SDC-based strand displacement and for experimental advice, Tristan Stérin for thoughts on the SDC model, Dave Doty for helpful comments, and Constantine Evans, David Soloveichik and Erik Winfree for insightful algorithmic discussions.

1 Introduction

Efficient algorithms play a crucial role in the design, analysis and engineering of nucleic acid systems including DNA computers and DNA nanostructures. Some decades ago the beautiful relationship between pseudoknot-free nucleic acid secondary structures[1] and dynamic programming algorithms was established [28, 17, 33]. For just one or more DNA/RNA strands, there are asymptotically exponentially many (in number of bases) distinct secondary structures those strands may adopt. Despite this, for the case of a single DNA or RNA strand, dynamic programming can be used to efficiently, in polynomial time, predict important global properties of the system, such as its pseudoknot-free minimum free energy (MFE) or partition function. Intuitively, the MFE is the energy of the most favoured[2] structure(s) of the system, and the partition function is the sum of the Boltzmann-weighted energy of each secondary structure of the system.

Background. The free energy $\Delta G(S) = \sum_{l \in S} \Delta G(l) + (n-1)\Delta G^{\text{assoc}}$ of an n-strand secondary structure S is a sum of empirically-obtained [22] free energies $\Delta G(l)$ of a set of *loop* features of the secondary structure such as stack loops, bulge loops, and many others [7]. $\Delta G^{\text{assoc}} > 0$ is an entropic penalty for bringing strands together. The MFE of a set Ω of unpseduoknotted secondary structures is simply $\min_{S \in \Omega} \Delta G(S)$, and the partition function is a (typically large positive) number $Q = \sum_{S \in \Omega} e^{-\Delta G(S)/k_{\text{B}}T}$ where k_{B} is Boltzmann's constant in units of $\text{kcal}/(\text{mol} \cdot \text{K})$ and T is temperature in Kelvin. Q is typically used as a normalisation factor to calculate the probability of any pseudoknot-free secondary structure S at equilibrium: $p(S) = (e^{-\Delta G(S)/k_{\text{B}}T})/Q$.

Dynamic-programming based MFE and partition function algorithms provide a firm basis for some sophisticated DNA sequence design tasks. For example, we might use partition function calculations in a design feedback loop with the goal of lowering the probability of unwanted ("off-target") secondary structures, and we may use MFE and/or partition function to improve the design of desired ("on-target") secondary structures [31, 10, 3, 7, 20].

As noted, there are fast MFE and partition function algorithms for systems consisting of a single strand. But for the case of multiple interacting strands, even unpseudoknotted, the situation is more nuanced: if we have a constant number (independent of input length) of strands, there is a polynomial time dynamic programming partition function algorithm [7], although, somewhat surprisingly, we don't know of one for MFE [4], since there are over-counting complications when there are multiple copies of the same strand. But if the system is multistranded in the most general sense, where the number of strands is given as part of the input (i.e. non-constant), unpseudoknotted MFE was recently shown to be both NP-hard and hard to approximate (APX-hard) and so is unlikely to have a polynomial time algorithm, even for a simple energy model (partition function is unknown for this case) [4]. Furthermore, if pseudoknots are permitted, MFE determination becomes NP-hard for a large class of reasonable energy models, even for a single strand [1, 15].

[1] We have an ordered list of DNA strands, where each strand is a sequence of DNA bases ordered in 5' to 3' direction, giving a total order on all DNA bases in the system. Then, a secondary structure is simply a set of pairs of the form (i, j) interpreted as "base i is paired with base j" (by convention $i < j$). We may permute the order of the strands, and a structure is called unpsedoknotted if there exists a strand order where such base pairs "do not cross", i.e. a strand ordering where there are no two pairs (i, j) and (i', j') such that $i' \in [i, \ldots, j]$ and $j' \notin [i, \ldots, j]$. This point is more intuitively explained using polymer graphs (see Figure 1).

[2] In physics and chemistry, more negative free energy generally means more favoured.

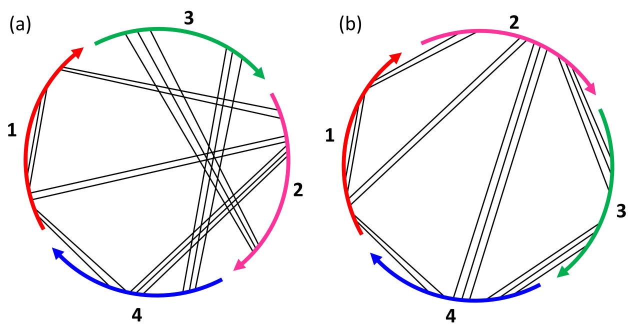


Figure 1 (a) Polymer graph for a secondary structure over the strand set $\{1, 2, 3, 4\}$ with strand ordering 1324. Some crossings are shown. (b) By simply reordering to 1234 we get another polymer graph for the same strand set and secondary structure, but without crossings. The existence of a strand ordering that yields a crossing-free diagram implies that this secondary structure is unpseudoknotted.

Despite the theoretical road-block of algorithmic hardness, multistranded and pseudoknotted self-assembly systems are some of the most successful DNA nanostructure and DNA computing paradigms: examples include DNA origami [21], RNA origami [12], and single-stranded tile systems [29, 31]. In these cases the design process has not been via full de novo algorithmic simulation of the thermodynamics, but by other methods including decomposition into smaller pieces more amenable to analysis [31], or by intuition-based whiteboard drawings, or some other method. Another key point is that such systems, and many others, are typically designed at a domain level of abstraction, where we ignore the details of DNA sequences and imagine a set of strands, each with one or more domains, and where several domains are imagined to have identical free energy, and domains that should not bind never in fact bind.

Motivation. But perhaps many engineered multistranded and/or pseudoknotted systems make use of design concepts, such as modularity, abstraction, principled thermodynamics, or some other intuitively-attractive principles that make their design and analysis algorithmically tractable? In other words, despite the above-cited algorithmic hardness results, we may ask if there are sub-classes of multistranded and/or pseudoknotted DNA systems, that on the one hand are interesting (experimentally implementable, make complex nanostructures, are capable of computation, etc.) yet on the other hand allow for efficient, e.g. polynomial time, MFE and partition function algorithms? Digging further into a computational nuance: since the ability to calculate MFE and/or partition function seems to require detailed (combinatorial/ enumerative) knowledge of a system one may wonder if there is a hierarchy of nucleic acid computing systems, classified according to difficulty of computing MFE and/or partition function, where higher levels corresponds to more expressive molecular computers capable of stronger computation than lower-level systems?

Scaffolded DNA Computer. Here we look at a specific multistranded system. Stérin, Eshra and Woods [24] introduced a thermodynamically favoured [8] model of computation called the Scaffolded DNA Computer (SDC). The model is computationally expressive and has theoretical thermodynamic benefits over most forms of molecular computing. In one dimension (1D SDC) the model is capable of simulating finite state automata and transducers, and in 2D simulates arbitrary Boolean circuits [5] (but proved for a slightly different model).

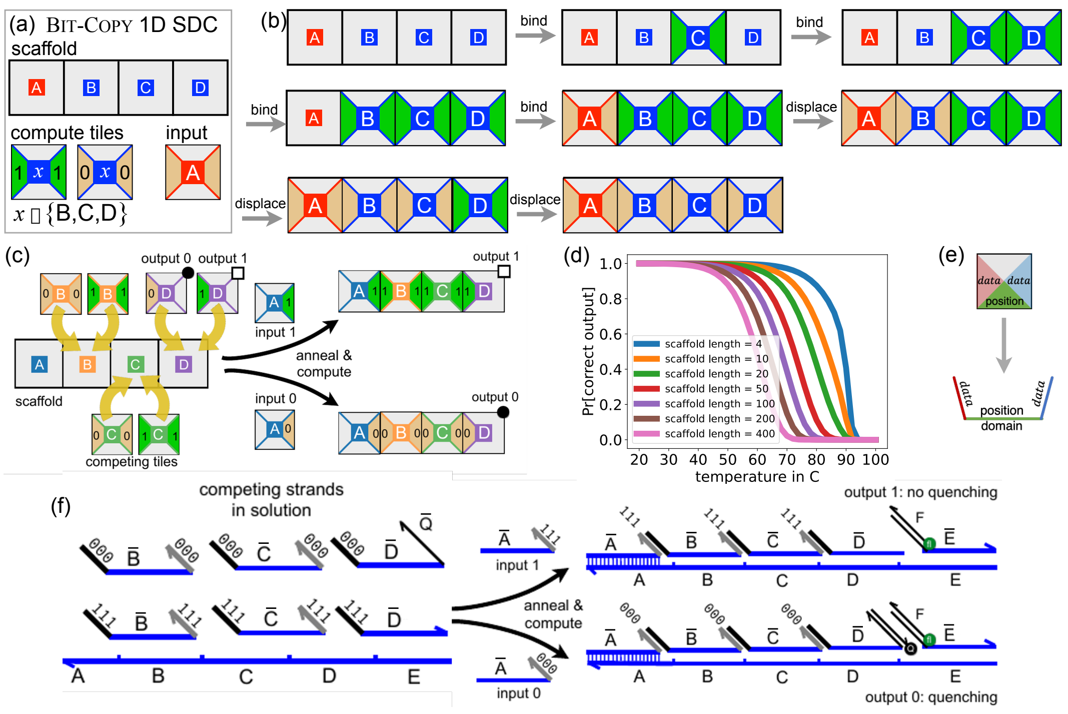


Figure 2 Thermodynamic model of the 1D scaffolded DNA computer (SDC) illustrated using a simple BIT-COPY program (a wire) of length $N = 4$ and maximum tiles per scaffold position $k = 2$. (a) Example BIT-COPY system for a length $N = 4$ scaffold. The system consists of 7 tiles: 2 per scaffold position B, C and D and 1 input tile at scaffold position A. (b) An example execution: starting from the empty scaffold, a tile may *bind* or *displace* an existing tile: in both cases the total number of matching bonds (scaffold-tile or tile-tile) should monotonically increase. 5 *bind* steps followed by 2 *displace* steps are shown. The final configuration has 4 scaffold-tile bonds and 3 tile-tile bonds, making it the most favourable. (c) Depending on whether input 0 or 1, the BIT-COPY system eventually reaches the most favoured (all-0 or all-1) configuration. (d) Using our partition function algorithm to simulate an anneal: simulations, for various scaffold lengths showing probability of target (correct output for BIT-COPY) with respect to temperature, using experimentally-relevant parameters, see Appendix E. (e) Schema for converting tiles to domain-level DNA strand abstraction. (f) Domain level abstraction of BIT-COPY system. 10 programs have been experimentally implemented in this model [24], the strand diagrams show some implementation features (3-bit domains for data and computing, as well as fluorescence/quencher reporting complexes).

Figure 2(a–c) describes the model at an abstract tile-level of abstraction suited to programming and theoretical analysis (see Section 2.1 for a domain-level model definition). This computational model borrows key principles from DNA origami [21] (low concentration scaffold, high concentration strands that bind to the scaffold), but breaks one origami principle by allowing non-scaffold strands to bind to each other, albeit somewhat weakly (no stronger than scaffold-tile binding). These principles yield a thermodynamically favoured model of molecular computation where the target (output) structure, is most favoured.

The SDC has been implemented experimentally [24] in 1D, showcasing a total of 10 programs that solve problems such as ADDITION of two 4-bit numbers, PARITY of an 8-bit input (is the number of 1s odd?), and GRAPH REACHABILITY (is there a path in an input graph from a source node s to a target node t). One future aim is to scale up to significantly larger 1D and 2D systems, by exploiting the thermodynamic favourability of the scaffolded design principle. However, to do that we need new algorithmic design tools since these systems are multistranded in 1D and 2D, and pseudoknotted in 2D.

Main results. We prove a number of results for the domain-based 1D SDC model. The following two theorems are respectively proven in Sections 3 and 4 (see Figure 2 for N, k).

▶ **Theorem 1.** There is an $O(k^2N)$ time algorithm to determine the domain-level MFE for a 1D SDC of length N with $\leq k$ computation strands competing at each scaffold domain.

▶ **Theorem 2.** There is an $O(k^2N)$ time algorithm for the domain-level partition function for a 1D SDC of length N with $\leq k$ computation strands competing at each scaffold domain.

Our third result opens a door to the research direction of classifying the complexity of (computational or not) nucleic acid systems based on the computational complexity of computing MFE/partition function for those systems (proof in Appendix D):

▶ **Theorem 3.** Given a 1D SDC $\mathcal{S}$ of length $N \in \mathbb{N}$, with $k = O(1)$ computing strands per scaffold domain and bounded domain energies ($\Delta G(d) \in O(1)$, for a domain d binding to its complement), and a finite-precision decimal r, the problem of deciding whether the domain-level MFE of $\mathcal{S}$ is $\leq r$ is in the complexity class L.

Our fourth contribution is exploratory: In Section 5 we give a stochastic kinetic simulator for 1D SDC systems. We evaluate the kinetics of the 1D SDC, and go on to test two strategies to speed up kinetics, the first using extra components and the second exploiting varying concentrations, both showing some speed-up. This initiates the kinetic study of the 1D SDC model, where one imagines operating the system at a fixed temperature allowing the system to move towards equilibrium via designed kinetic pathways. This stands in contrast to previous work [24] which focused on using the 1D SDC thermodynamically: heat it up, cool it down!

1.1 Related work

The SDC leverages thermodynamic favourability principles from DNA origami [21], and computational abilities from self-assembly [31] and strand displacement systems such as the scaffolded SIMD||DNA system [26, 27] as well as theory-based systems [8]. We wish to create computational tools to evaluate future strategies for both thermodynamic computation, and constant-temperature (kinetic) computation. But the SDC has a unique combination of self-assembly and strand displacement features that make direct use of existing tools problematic and led us to design bespoke thermodynamic (MFE and partition function) prediction algorithms, and a kinetic simulator, all built on the same energy model.

SDC self-assembly features include: growth of large target structures, a large combinatorically-described set of intermediate structures, hassle-free scaffold-mediated seeded growth with in-built avoidance of unwanted off-scaffold nucleation (computational domains bind weakly, hence enumeration/simulation of *all* complexes is not desired nor even relevant). SDC strand-displacement features include: use of toeholds, and displacement domains which are sometimes desired to work along standard toehold-mediated strand displacement style pathways, but other times not, *depending on our choice of operating mode* – so far experimental work used a simple, fast anneal [24], without needing careful kinetic control!

Related work on MFE and partition function was already discussed above. Multistranded MFE is NP-hard [4], a result that holds in a simple domain-level model. Our polynomial time MFE algorithm (Theorem 1) is also for a multistranded domain-level system; however, we exploit the 1D SDC binding structure to side-step the general-case hardness result of [4].

Related work on kinetic simulators. There are a variety of simulators suited to strand-displacement systems and self-assembly based systems. One goal is to scale up to large scaffold lengths, both in 1D and 2D (with pseudoknots), which may require new features not seen in existing fine/coarse-grained strand displacement simulators [25, 2, 19, 23, 32]. Also, our use of toehold mediated strand displacement, as well as future goals to handle displacement through helices (in 2D), require features that go beyond existing self-assembly simulators [30, 9, 11]. Our kinetic model uses rates from the peppercorn reaction enumerator [2]. In future work we could use oxDNA [25] to further inform and calibrate rates.

1.2 Future work

Are there fast thermodynamic prediction algorithms for other engineered multistranded and/or pseudoknotted systems? Specifically, for DNA strand displacement circuits, or DNA tile-based self-assembly systems, or even DNA origami systems? Our main motivation for this question, and in fact for this work, is the idea that designed systems and abstractions might side-step known hardness results and imply the existence of efficient MFE or partition function algorithms. Here, for 1D SDC, two things helped: (1) using a domain-based model, justified by enforcement of domain-level abstraction/interactions with careful sequence design, and (2) that the set of all 1D SDC configurations have an implicit structure amenable to divide-and-conquer, despite that set being exponentially large in system size.

Continuing this reasoning a step further, one may ask, beyond the existence of a fast, polynomial time, thermodynamic prediction algorithm, can one attempt to classify engineered DNA systems based on the computational complexity of their MFE or partition function? Theorem 3 shows that MFE for 1D SDC is in the complexity class L(with the assumption $k = O(1)$; we leave it open to remove/generalise that caveat). Hence, we can say that if we encode the output of a computation in an MFE configuration then the model solves only problems in L. We know that (general) multi-stranded systems are NP-complete MFE [4]. Here, we are asking if there are natural systems with MFE that lie in other complexity classes, for example perhaps the wide variety of engineered domain-based DNA systems have algorithmic hardness of MFE that are characterised by some of the many complexity classes within P [16, 14].

More concretely, it remains open to characterise the complexity of predicting, reasonable domain-level formalisations of, 2D SDC systems.

2 1D Scaffolded DNA Computer (SDC): model definition

Previous work [24] defined the SDC at tile, domain, and sequence levels of abstraction, but in a mainly experimental context. In this section we formally define the 1D SDC at the domain level of abstraction, with sufficient extra technical detail to facilitate our proofs.

2.1 A simple domain-based model of 1D SDC

▶ **Definition 4** (domain, complement, strand, binding)**.** A *domain* d is an ordered triple $(\text{name}, \text{len}, \text{dir})$, where name is a string over a fixed alphabet, $\text{len} \in \mathbb{N}$ is the length of d, and $\text{dir} \in \{\rightarrow, \leftarrow\}$ is the domain direction. The complement of domain d is the equal-length opposite-direction domain $\bar{d}$. An n-domain strand s, is an ordered n-tuple of domains $s = (d_1, d_2, ..., d_n)$ all of the same direction (the strand is said to have that direction). We say $d_i < d_j$ if $i < j$. Two strands s_1 and s_2 may bind at domain d, if one has domain d and the other $\bar{d}$.

▶ **Definition 5** (1D Scaffolded DNA Computer (SDC)). A 1D Scaffolded DNA Computer (Figure 2), $\mathcal{S}$, is an ordered pair (S, T) where $S = (d_1, d_2, \ldots d_N)$ is an N-domain *scaffold strand* with direction $\leftarrow$, with all N *scaffold domains* being distinct. T is a set of 3-domain *computation strands*, of direction $\rightarrow$, and each $s \in T$ is the form $s = (d^{\mathrm{L}}, d_i^{\mathrm{M}}, d^{\mathrm{R}})$ where $d^{\mathrm{L}}, d_i^{\mathrm{M}}, d^{\mathrm{R}}$ are the domains on left, middle and right of s respectively, and where d_i^{M} binds to a scaffold domain d_i and $d^{\mathrm{L}}, d^{\mathrm{R}}$ are called *computation domains* and they do not bind to any scaffold domain. $\mathcal{S}$ is said to be of length N.

We say a computation strand s *competes* at scaffold domain d_i if its middle domain $d_i^{\mathrm{M}} = \bar{d}_i$. The set of d_i*-competing strands* is $C_{d_i} = \{s \mid s = (d^{\mathrm{L}}, d_i^{\mathrm{M}}, d^{\mathrm{R}}) \in T \text{ and } d_i^{\mathrm{M}} = \bar{d}_i\}$. Since scaffold domains are distinct, if $d_i \neq d_j$ then $C_{d_i} \cap C_{d_j} = \emptyset$ (hence each computation strand binds at most at one scaffold domain). Sometimes, the first scaffold domain, d_1, is called the input domain. For any two computation strands $s \in C_{d_i}$ and $s' \in C_{d_j}$, we say $s \prec s'$ if $d_i < d_j$.

▶ **Definition 6** (SDC configuration, SDC secondary structure). For a 1D SDC $\mathcal{S}$ of length N, a *configuration* X is a sequence of $l \leq N$ computation strands $X = (s_1, s_2, \ldots, s_l)$ such that $s_i \prec s_{i+1}$. X is said to be of size l. We interpret a configuration as a *domain-level secondary structure* by binding up all adjacent matching domains, more formally for all $i \in [1, l]$:

(a) we bind the middle domain d_i^{M} of s_i to s_i's associated scaffold domain (since $s_i \in C_d$ for some d), and

(b) if $d_i^{\mathrm{R}} = \bar{d}_{i+1}^{\mathrm{L}}$ and s_i, s_{i+1} are adjacent on the scaffold, i.e. if the right domain of s_i is complement to the left of s_{i+1} and if the scaffold indices of s_i, s_{i+1} differ by exactly 1, we bind those two domains.

In the previous definition, the condition $s_i \prec s_{i+1}$ ensures that a configuration has at most one competing computation strand per scaffold domain.

A *computation* is a sequence of configurations $\epsilon \vdash X_1 \vdash X_2 \vdash \ldots \vdash X_t$ that begins with the empty configuration ϵ (corresponding to a scaffold with nothing bound to it). A computation step, $X_j \vdash X_{j+1}$, is where a computation strand s binds at some scaffold domain d, where s is non-deterministically chosen from those d-competing strands C_d that preserve or increase the total number of bound domains.[3] A configuration X_j is *final* if X_j has the maximum number of bound domains out of all possible configurations. A final configuration may be unique or not.[4]

Figure 2(c) shows an initial configuration (left), and two final configurations (right). A useful intuition for computation in this model: the selection of a strand (from the finite set of possibilities for that position) executes a logical step, and step-by-step the system will moves towards an enthalpically favourable reachable configuration (or a cycle of them).

2.2 Ensemble of secondary structures

An MFE or partition function algorithm computes over an ensemble (i.e. a set) of permissible secondary structures. In this work, we choose that ensemble to be the set of domain-level secondary structures, that each include a scaffold strand with zero or more computation strands bound to that scaffold and also to each other, or more formally:

[3] Note that off-scaffold interactions are forbidden in this formalism. In Section 2.3 we will define a free-energy based condition that more closely matches experimental implementation.

[4] In the Bit-Copy example in Figure 2 the system eventually reaches one of two polymers (all 0 or all 1), depending on the input (0 or 1). Other example SDC systems have multiple equal-energy polymers, and there may or may not be a cycle (of length > 1) of steps between elements of a set of polymers. A cycle can happen where there are several polymers (each with an identical number of unbound domains) and the system transitions between these.

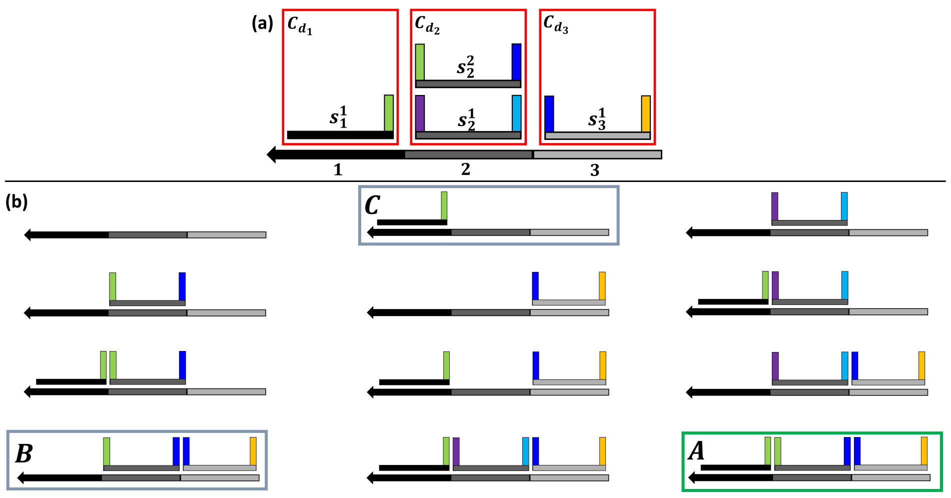


Figure 3 A small example 1D SDC and its ensemble of configurations. (a) An example 1D SDC $\mathcal{S}$ of length 3 showing its scaffold and four computation strands. At scaffold positions 1, 2 and 3 there are respectively 1, 2 and 1 competing computation strands. (b) The ensemble of configurations $\Omega^{\mathcal{S}}$, with A highlighted as the unique MFE configuration of this system (bound domains everywhere).

▶ **Definition 7** (ensemble $\Omega^{\mathcal{S}}$). The ensemble $\Omega^{\mathcal{S}}$ of a 1D SDC $\mathcal{S}$ is the set of all configurations of $\mathcal{S}$ interpreted as domain-level secondary structures (see Definition 6).

For a 1D SDC $\mathcal{S}$ of length N that has $k_1, k_2, ..., k_N$ computation strands competing at the N scaffold domains, we have $|\Omega^{\mathcal{S}}| = (k_1+1)(k_2+1)\cdots(k_N+1)$ since each scaffold domain d_i has k_i possible strands that bind it plus the possibility of it being unbound. If there are exactly k competing strands at each scaffold domain, $|\Omega^{\mathcal{S}}| = (k+1)^N$.

▶ **Example 8.** Figure 3 shows a simple 1D SDC of length 3 and its ensemble of twelve configurations/domain-level secondary structures.

Our choice of secondary structure ensemble is justified as follows: We ignore off-scaffold interactions by making the assumption that the 1D SDC system is designed so that (a) scaffold and computational strands do not self-bind, (b) any pair of matching computation domains are sufficiently weak that a pair of 3-domain strands are unlikely to bind for much time (even if they bind on two computation domains), (c) by using a principle analogous to DNA origami [21], we assume the scaffold is at low concentration relative to the computation strands and that (d) scaffold binding domains are strong ensuring that binding to the scaffold is highly favoured.

It can be seen that the domain-level secondary structures we consider are unpseudoknotted. This fact is not required to prove our theorems, so we omit (the straightforward) proof.

2.3 Energy model

We augment each domain d to have an associated negative real-valued free energy $\Delta G(d) \in \mathbb{R}, \Delta G(d) < 0$. Also, we define ΔG^{assoc} to be a strictly positive real number such that $-\Delta G(d) > \Delta G^{\text{assoc}}$. For notation, we let $\Delta G(d, d') = \Delta G(d)$ if $d = \bar{d}'$, and $\Delta G(d, d') = 0$ otherwise.

For a 1D SDC $\mathcal{S}$ of length N, the free energy of a configuration $X \in \Omega^{\mathcal{S}}$ of size l is:

$$\Delta G^{\mathcal{S}}(X) = \sum_{s\in X} \Delta G(d^{\text{M}}(s)) + l \cdot \Delta G^{\text{assoc}} + \sum_{s_i, s_{i+1}\in X} \Delta G(d^{\text{R}}(s_i), d^{\text{L}}(s_{i+1})). \quad (1)$$

where the latter summation sums the free energy of bound *scaffold-adjacent* strand pairs. Note that since X represents $l+1$ strands *including the scaffold*, the strand association penalty of $l \cdot \Delta G^{\text{assoc}}$ is consistent with previous work on multiple strands, e.g. [7, 4]). The 1D SDC domain-level MFE and partition function are respectively:

$$\text{MFE}^{\mathcal{S}} = \min_{X \in \Omega^{\mathcal{S}}} \{\Delta G^{\mathcal{S}}(X)\} \tag{2}$$

$$Q^{\mathcal{S}} = \sum_{X \in \Omega^{\mathcal{S}}} e^{-\Delta G^{\mathcal{S}}(X)/k_{\text{B}}T} \tag{3}$$

▶ **Example 9.** The free energy of configuration B in Figure 3 is:

$$\Delta G^{\mathcal{S}}(B) = \Delta G(d^{\text{M}}(s_2^2)) + \Delta G(d^{\text{M}}(s_3^1)) + 2\Delta G^{\text{assoc}} + \Delta G(d^{\text{R}}(s_2^2), d^{\text{L}}(s_3^1))$$

Configuration A in Figure 3 has MFE of all configurations, $\text{MFE}^{\mathcal{S}} = \Delta G^{\mathcal{S}}(A)$, since A has all scaffold domains bound, and all adjacent computation domains bound.

3 Thermodynamics: polynomial time 1D SDC MFE algorithm

Before proving the main result of this section, Theorem 1, we give some preliminary results.

3.1 Definitions for combining configurations & ensemble partitioning

▶ **Definition 10** ($X \circledast Y$)**.** For any two configurations X and Y such that $\forall s \in X, \forall s' \in Y$ such that $s \in C_d \implies s' \notin C_d$, we say X is compatible with Y. The interlacing of two compatible configurations X and Y is the configuration $X \circledast Y = (s_1, s_2, \ldots, s_k)$ such that $s_i \prec s_{i+1}$, and either $s_i \in X$ or $s_i \in Y$. The interlacing of a configuration X with the set of configurations Λ where X is compatible with Λ, is the set $X \circledast \Lambda = \{X \circledast Y \mid \text{ for each } Y \in \Lambda\}$.

▶ **Example 11.** Configuration B is compatible with configuration C in Figure 3 since they do not share any common, bound scaffold domain, $B \circledast C = A$.

The following definitions will be useful to efficiently partition the (exponentially large) SDC ensemble $\Omega^{\mathcal{S}}$ to compute its MFE and partition function.

▶ **Definition 12** ($\Omega_s^{\mathcal{S}}$)**.** For a 1D SDC $\mathcal{S}$, $\Omega_s^{\mathcal{S}}$ denotes the set of all configurations having the computation strand s as the (rightmost) strand, more formally, $\Omega_s^{\mathcal{S}} = \{X \mid X \in \Omega^{\mathcal{S}} \text{ and } s \in X \text{ and } \forall s' \in X \text{ such that } s' \neq s \implies s' \prec s\}$.

The intuition behind Lemma 13 comes from: if we have any configuration with final strand s, we can see it as the interlacing of two smaller configurations one of the two is just $\{s\}$ itself and the other ends with a final strand $s' \prec s$.

▶ **Lemma 13.** *In any 1D SDC $\mathcal{S}$, for any computation stand s, $\Omega_s^{\mathcal{S}} = [s \circledast \bigcup_{s' \prec s} \Omega_{s'}^{\mathcal{S}}] \bigcup \{s\}$ and $|\Omega_s^{\mathcal{S}}| = 1 + \sum_{s' \prec s} |\Omega_{s'}^{\mathcal{S}}|$.*

▶ **Definition 14** (Ω-associated MFE)**.** For a set of configurations $\Omega \subset \Omega^{\mathcal{S}}$ of a 1D SDC $\mathcal{S}$, the Ω-associated MFE, $\text{MFE}^{\mathcal{S}}(\Omega) = \min_{X \in \Omega}\{\Delta G^{\mathcal{S}}(X)\}$. If $\Omega = \Omega_s^{\mathcal{S}}$ for some computation strand s, then for simplicity we will denote $\text{MFE}^{\mathcal{S}}(\Omega_s^{\mathcal{S}})$ by $\text{MFE}_s^{\mathcal{S}}$.

▶ Remark 15. To simplify our proofs, from Observation 17 onwards, for any scaffold domain d, we assume that $|C_d| \neq 0$, meaning there is at least one computation strand competing at d. At the end of this section (Remark 20) we will show how our proposed algorithm can be easily used to overcome this restriction.

From Section 2.3, (a) all domain binding energies are negative, and (b) $\Delta G(s) < -\Delta G^{\text{assoc}}$ for a scaffold-bound strand s. Also, for a configuration Y of length $l < N$ we can add $N - l$ computation strands to the $N - l$ unbound scaffold positions, lowering the free energy (this fact is used in the following two lemmas). The following lemma follows from the inclusion $\Omega^{\mathcal{S}}_{s'} \subset \Omega^{\mathcal{S}}_{s}$:

▶ **Lemma 16.** *In any 1D SDC $\mathcal{S}$, for any two different computation strands s and s' such that if $s' \prec s$, then $MFE^{\mathcal{S}}_{s'} > MFE^{\mathcal{S}}_{s}$.*

▶ **Observation 17.** *For a 1D SDC $\mathcal{S}$ of length N, there is a configuration X of size N such that $MFE^{\mathcal{S}} = \Delta G^{\mathcal{S}}(X)$.*

▶ **Lemma 18.** *In any 1D SDC $\mathcal{S}$, for any two different computation strands s and s' such that if $s' \prec s$, then $MFE^{\mathcal{S}}(\Omega^{\mathcal{S}}_{s'} \circledast \{s\}) = MFE^{\mathcal{S}}(\Omega^{\mathcal{S}}_{s'}) + \Delta G^{\mathcal{S}}(d^{\text{M}}(s)) + \Delta G^{\text{assoc}} + \Delta G(d^{\text{R}}(s'), d^{\text{L}}(s))$.*

3.2 Polynomial time 1D SDC MFE algorithm

The main result of this section is the following (restated) theorem:

▶ **Theorem 1.** There is an $O(k^2N)$ time algorithm to determine the domain-level MFE for a 1D SDC of length N with $\leq k$ computation strands competing at each scaffold domain.

Proof. We will prove that Algorithm 1, intuitively illustrated in Appendix A, returns the recursively-defined quantity $M^{\mathcal{S}}$ defined in Equation (4), and does so in $O(k^2N)$ steps (we assume the standard RAM model [6] for algorithm analysis and $\Delta G = O(1)$ per domain).

$$M^{\mathcal{S}} = \min_{s \in C_{d_N}} \{M^{\mathcal{S}}_s\} \tag{4}$$

$$M^{\mathcal{S}}_s = \Delta G(d^{\text{M}}(s)) + \Delta G^{\text{assoc}} + \min_{s' \in L_s} \{M^{\mathcal{S}}_{s'} + \Delta G(d^{\text{R}}(s'), d^{\text{L}}(s))\} \tag{5}$$

where C_{d_N} is the set of strands competing at the final/rightmost scaffold domain (Section 2.1), L_s is the set of strands that bind to the domain immediately "to the left" of s on the scaffold (i.e. if $s \in C_{d_i}$ then $L_s = C_{d_{i-1}}$, and where $L_s = \emptyset$ if $s \in C_{d_1}$), and the remaining notation is given in Section 2.3.

To prove the time bound: The for loops in Lines 3 and 5 iterate over all strands s in the system, iteratively implementing the recursion in Equation (5), with each iteration (for s) representing a corresponding L_s (Equation (5)). Then the min over s' (Equation (5)) is executed by Line 10 (as a min over m). There are $\leq Nk$ strands, and the min requires $O(k)$ steps, giving $O(Nk^2)$ time. Finally, Line 14 executes Equation (4) in an additional $O(k)$ steps, yielding the claim.

It remains to show that $M^{\mathcal{S}}$ (Equation (4)) equals $\text{MFE}^{\mathcal{S}}$ (see Equation (2)): By Lemma 19 (below), for any computation strand s, $M^{\mathcal{S}}_s = \text{MFE}^{\mathcal{S}}_s$, thus it is immediate that the same strand s satisfies both $\min_{s \in T} M^{\mathcal{S}}_s = \min_{s \in T} \text{MFE}^{\mathcal{S}}_s$. Since an MFE configuration has size N, there is some $s \in C_{d_N}$, based on the assumption in Remark 15, such that $\text{MFE}^{\mathcal{S}}_s = \text{MFE}^{\mathcal{S}}$, and hence for that s, $M^{\mathcal{S}}_s = \text{MFE}^{\mathcal{S}}$. ◀

▶ **Lemma 19.** *For any 1D SDC $\mathcal{S}$, for any computation strand s, $M^{\mathcal{S}}_s = MFE^{\mathcal{S}}_s$ (where $M^{\mathcal{S}}_s$ is from Equation (5) and $MFE^{\mathcal{S}}_s$ is from Definition 14).*

Proof. We will prove the statement by induction on x, such that x is the domain index where the computation strand s is competing. Formally if $s \in C_{d_i}$, then $x = i$.

Algorithm 1 1D SDC MFE algorithm. The proof of Theorem 1 proof shows that this algorithm returns the value $M^{\mathcal{S}}$ defined in Equation (4). Note that arrays are indexed from 1. Notation: $k_1, \ldots, k_N$ are the counts of competing strands at scaffold domains $d_1, \ldots, d_N$. Let s_i^j be the j^{th} strand competing at domain d_i.

```
1:  $M_{\text{curr}} = [0, 0, \ldots, 0]$                ▷ size $k = \max(k_1, \ldots, k_N)$ for current MFEs
2:  $M_{\text{prev}} = [0, 0, \ldots, 0]$                ▷ size $k = \max(k_1, \ldots, k_N)$ for previous MFEs
3:  for $i \leftarrow 1 \ldots N$ do                     ▷ index scaffold domains
4:      $M_{\text{prev}} \leftarrow M_{\text{curr}}$
5:      for $j \leftarrow 1 \ldots k_i$ do               ▷ index computational strands at scaffold domain $d_i$
6:          if $i = 1$ then                              ▷ first scaffold domain, has no left neighbour
7:              $M_{\text{curr}}[j] \leftarrow \Delta G(d^{\text{M}}(s_i^j))$
8:          else
9:                       ▷ $O(k)$ steps to choose min and bind scaffold + entropic penalty
10:             $M_{\text{curr}}[j] \leftarrow \left[\min_{m \in \{1,2,\ldots,k_{i-1}\}} \left(M_{\text{prev}}[m] + \Delta G\left(d^{\text{R}}(s_{i-1}^m), d^{\text{L}}(s_i^j)\right)\right)\right.$
                        $\left. + \Delta G(d^{\text{M}}(s_i^j)) + \Delta G^{\text{assoc}}\right]$
11:         end if
12:     end for
13: end for
14: $M^{\mathcal{S}} \leftarrow \min_{k' \in \{1,2,\ldots,k_N\}} M_{\text{curr}}[k']$     ▷ $O(k)$ steps implement Equation (4) giving $M^{\mathcal{S}}$
15: return $M^{\mathcal{S}}$
```

Base step: if $x = 1$, then this means that s is a strand competing at the first scaffold domain, then using Lemma 13 there is only one configuration $X \in \Omega_s^{\mathcal{S}}$, and $X = s$. Then we have $M_s^{\mathcal{S}} = \Delta G(d^{\text{M}}(s)) + \Delta G^{\text{assoc}} = \Delta G^{\mathcal{S}}(X) = \text{MFE}_s^{\mathcal{S}}$, and this completes the base step.

Induction step: assume the induction hypothesis is valid for all $x < y$, in other words that $M_{s'}^{\mathcal{S}} = \text{MFE}_{s'}^{\mathcal{S}}$ (the former from Equation (5), the latter from Definition 14) because x, y are domain indices and $s' \prec s$.

Then, if $x = y$, we can replace $M_{s'}^{\mathcal{S}}$ in Equation (5) by $\text{MFE}_{s'}^{\mathcal{S}}$, we get the following: $M_s^{\mathcal{S}} = \Delta G(d^{\text{M}}(s)) + \Delta G^{\text{assoc}} + \min_{s' \in L_s}\{\text{MFE}_{s'}^{\mathcal{S}} + \Delta G(d^{\text{L}}(s),\, d^{\text{R}}(s'))\}$. As $\Delta G(d^{\text{M}}(s)) + \Delta G^{\text{assoc}}$ is just a constant, and using the basic properties of the min operator, we get the following: $M_s^{\mathcal{S}} = \min_{s' \in L_s}\{\text{MFE}_{s'}^{\mathcal{S}} + \Delta G(d^{\text{M}}(s)) + \Delta G^{\text{assoc}} + \Delta G(d^{\text{L}}(s),\, d^{\text{R}}(s'))\}$. Using Lemma 18, we will get the following: $M_s^{\mathcal{S}} = \min\limits_{s' \in L_s}\{\text{MFE}^{\mathcal{S}}(\Omega_{s'}^{\mathcal{S}} \circledast \{s\})\}$. Using Definition 14, we get: $M_s^{\mathcal{S}} = \min\limits_{s' \in L_s}\{\min\limits_{X \in (\Omega_{s'}^{\mathcal{S}} \circledast \{s\})}\{\Delta G^{\mathcal{S}}(X)\}\}$. For each distinct computation strand s', it is easy to show that each set of configurations $[\Omega_{s'}^{\mathcal{S}} \circledast \{s\}]$ is disjoint. Let $\Omega' = \bigcup\limits_{s' \in L_s} [\Omega_{s'}^{\mathcal{S}} \circledast \{s\}]$, so again we use another property of the min operator to get the following: $M_s^{\mathcal{S}} = \min\limits_{X \in \Omega'}\{\Delta G^{\mathcal{S}}(X)\}$. From Observation 17 we know that if Y is the configuration that has the MFE of the partition $\Omega_s^{\mathcal{S}}$, then Y must be of length l if $s \in C_{d_l}$, in other words Y must have computation strands only on scaffold domains $0, 1 \ldots, l$. Any configuration of size l that ends in s is contained in $\Omega_s^{\mathcal{S}}$, and is also contained in Ω', and hence $Y \in \Omega'$, and we get that $M_s^{\mathcal{S}} = \min\limits_{X \in \Omega'}\{\Delta G^{\mathcal{S}}(X)\} = \min\limits_{X \in \Omega_s^{\mathcal{S}}}\{\Delta G^{\mathcal{S}}(X)\}$. So $M_s^{\mathcal{S}} = \min\limits_{X \in \Omega_s^{\mathcal{S}}}\{\Delta G^{\mathcal{S}}(X)\} = \text{MFE}_s^{\mathcal{S}}$. ◀

▶ Remark 20. Now, we can easily remove the assumption in Remark 15 by dividing the scaffold into pieces separated by domains d such that $C_d = \emptyset$ and running MFE on each such region, and then summing each MFE.

Finally, Appendix D gives an analysis of Algorithm 1 that yields the proof of Theorem 3.

Algorithm 2 1D SDC partition function algorithm. The proof of Theorem 2 argues that this algorithm returns $Z^{\mathcal{S}}$ as defined in Equation (6). Note that arrays are indexed from 1, and recall that $k_1, \ldots, k_N$ are the counts of competing strands at scaffold domains $d_1, \ldots, d_N$, and we let s_i^j be the j^{th} strand competing at domain d_i. See Figure 9.

```
1: $Z_{\text{curr}} = [0, 0, \ldots, 0]$    ▷ size $k = \max(k_1, \ldots, k_N)$, current (partial) partition function
2: $Z_{\text{prev}} = [0, 0, \ldots, 0]$    ▷ size $k = \max(k_1, \ldots, k_N)$, previous (partial) partition function
3: $Z^{\mathcal{S}} \leftarrow 1$;    $\text{sum}_a \leftarrow 0$
4: for $i \leftarrow 1 \ldots N$ do
5:     $\text{sum}_a \leftarrow \text{sum}_a + \sum_{i \in \{1,\ldots,k\}} Z_{\text{prev}}[i]$    ▷ $\text{sum}_a$: rightmost summation Equation (7)
6:     $Z_{\text{prev}} \leftarrow Z_{\text{curr}}$
7:     $Z_{\text{curr}} = [0, 0, \ldots, 0]$
8:     for $j \leftarrow 1 \ldots k_i$ do    ▷ each iteration computes Equation (7) for a strand
9:         $t_1 = \mathrm{e}^{-(\Delta G(d^{\mathrm{M}}(s_i^j)) + \Delta G^{\text{assoc}})/k_{\mathrm{B}}T}$
10:        if $i = 1$ then    ▷ first domain where is no neighbors at all
11:            $Z_{\text{curr}}[j] = t_1$
12:        else
13:            $t_2 \leftarrow 0$
14:            for $m \leftarrow 1 \ldots k_{i-1}$ do
15:                $t_2 \leftarrow t_2 + \left(\mathrm{e}^{-\left(\Delta G(d^{\mathrm{R}}(s_{i-1}^m), d^{\mathrm{L}}(s_i^j))\right)/k_{\mathrm{B}}T}\right) \cdot Z_{\text{prev}}[m]$
16:            end for
17:            $Z_{\text{curr}}[j] \leftarrow t_1 + t_2 + \text{sum}_a$
18:        end if
19:        $Z^{\mathcal{S}} \leftarrow Z^{\mathcal{S}} + Z_{\text{curr}}[j]$    ▷ computing Equation (6)
20:    end for
21: end for
22: return $Z^{\mathcal{S}}$
```

4 Thermodynamics: polynomial time 1D SDC partition function

▶ **Definition 21** (Ω-associated partition function)**.** For a 1D SDC $\mathcal{S}$, for any set of configurations $\Omega \subset \Omega^{\mathcal{S}}$, the Ω-associated partition function is $Q^{\mathcal{S}}(\Omega) = \sum_{X \in \Omega} e^{-\Delta G^{\mathcal{S}}(X)/k_{\mathrm{B}}T}$. If $\Omega = \Omega_s^{\mathcal{S}}$ for some computation strand s, then for simplicity we will denote $Q^{\mathcal{S}}(\Omega_s^{\mathcal{S}})$ by $Q_s^{\mathcal{S}}$.

4.1 A polynomial time partition function algorithm for 1D SDC

We restate the main result of this section:

▶ **Theorem 2.** There is an $O(k^2N)$ time algorithm for the domain-level partition function for a 1D SDC of length N with $\leq k$ computation strands competing at each scaffold domain.

Proof. We will first claim that Algorithm 2, intuitively illustrated in Figure 9 (Appendix B), returns the recursively-defined quantity $Z^{\mathcal{S}}$ (Equation (6)), and does so in $O(k^2N)$ steps:

$$Z^{\mathcal{S}} = 1 + \sum_{s \in T} Z_s^{\mathcal{S}} \tag{6}$$

$$Z_s^{\mathcal{S}} = \left(e^{-(\Delta G(d^{\mathrm{M}}(s)) + \Delta G^{\text{assoc}})/k_{\mathrm{B}}T}\right) \cdot \left[1 + \sum_{s' \in L_s} Z_{(s',s)} * Z_{s'}^{\mathcal{S}} + \sum_{s' \prec s \text{ and } s' \notin L_s} Z_{s'}^{\mathcal{S}}\right] \tag{7}$$

where $Z_{(s',s)} = e^{-\Delta G(d^{\mathrm{R}}(s'),d^{\mathrm{L}}(s))/k_{\mathrm{B}}T}$. Here, $Z^{\mathcal{S}}_s$ is an intermediate quantity (that we will prove is the Ω_s-associated partition function), L_s is the set of strands that bind to the domain immediately "to the left" of s on the scaffold (i.e. if $s \in C_{d_i}$ then $L_s = C_{d_{i-1}}$, and where $L_s = \emptyset$ if $s \in C_{d_1}$).

To see the claim note that Lines 4 and 8 iterate over all $s \in T$ ($O(kN)$ iterations), and each inner-loop iteration calculates $Z^{\mathcal{S}}_s$ for some strand $s \in T$. Line 14 for loop iterates $O(k)$ times and computes the left-hand summation of Equation (7) (giving $O(k^2N)$ total steps), and Line 5 computes the right-hand summation. Line 19 accumulates the value of $Z^{\mathcal{S}}$ at each iteration, and when the algorithm terminates $Z^{\mathcal{S}}$ is equal to Equation (6).

Lemma 23 below proves that $Z^{\mathcal{S}} = Q^{\mathcal{S}}$, which completes the proof. ◀

▶ **Lemma 22.** *For any 1D SDC $\mathcal{S}$, for any computation strand s, $Z^{\mathcal{S}}_s = Q^{\mathcal{S}}_s$ (where $Z^{\mathcal{S}}_s$ is from Equation* (7) *and $Q^{\mathcal{S}}_s$ is from Definition 21)*

Proof. We will prove it by induction on scaffold domain index x, with the induction hypothesis being that $Z^{\mathcal{S}}_s = Q^{\mathcal{S}}_s$ for all s at scaffold domain index $\leq x \in [1, N]$.

Base step, $x = 1$: This implies that strand s is competing at first scaffold domain d_1, then using Lemma 13 there is only one secondary structure $X \in \Omega^{\mathcal{S}}_s$, hence $X = s$, a configuration of size 1. Then we compete the base step via:

$$Z^{\mathcal{S}}_s = \mathrm{e}^{-(\Delta G(d^{\mathrm{M}}(s)) + \Delta G^{\mathrm{assoc}})/k_{\mathrm{B}}T} = \mathrm{e}^{-(\Delta G^{\mathcal{S}}(X))/k_{\mathrm{B}}T} = Q^{\mathcal{S}}_s.$$

We re-write $Z^{\mathcal{S}}_s$ (Equation (7)) as follows:

$$Z^{\mathcal{S}}_s = \left(\mathrm{e}^{-(\Delta G(d^{\mathrm{M}}(s))+\Delta G^{\mathrm{assoc}})/k_{\mathrm{B}}T}\right) \cdot \left[1 + \sum_{s' \prec s} Z_{(s',s)} \cdot Z^{\mathcal{S}}_{s'}\right]. \tag{8}$$

which is equivalent to Equation (7), as $Z_{(s',s)} = 1$ if $s' \notin L_s$ since $\Delta G(d^{\mathrm{R}}(s'), d^{\mathrm{L}}(s))$ will equal 0 as s', s are not adjacent, i.e. there is no interaction between them.

Induction step: assume the induction hypothesis is valid for all $x < y$, then if $x = y$, we can replace $Z^{\mathcal{S}}_{s'}$ in Equation (8) by $\sum_{X \in \Omega^{\mathcal{S}}_{s'}} e^{-\frac{\Delta G^{\mathcal{S}}(X)}{k_{\mathrm{B}}T}} = Q^{\mathcal{S}}_s$ (Definition 21), we get

$$Z^{\mathcal{S}}_s = \left(\mathrm{e}^{-\frac{\Delta G(d^{\mathrm{M}}(s)) + \Delta^{assoc}}{k_{\mathrm{B}}T}}\right) \cdot \left[1 + \sum_{s' \prec s} Z_{(s,s')} \cdot \sum_{X \in \Omega^{\mathcal{S}}_{s'}} e^{-\frac{\Delta G^{\mathcal{S}}(X)}{k_{\mathrm{B}}T}}\right]. \tag{9}$$

Using the distributive property of multiplication over addition we get

$$Z^{\mathcal{S}}_s = \mathrm{e}^{-\frac{\Delta G(d^{\mathrm{M}}(s)) + \Delta G^{\mathrm{assoc}}}{k_{\mathrm{B}}T}} + \sum_{s' \prec s} \sum_{X \in \Omega^{\mathcal{S}}_{s'}} \left[\mathrm{e}^{-\frac{\Delta G(d^{\mathrm{M}}(s)) + \Delta G^{\mathrm{assoc}}}{k_{\mathrm{B}}T}} \cdot Z_{(s,s')} \cdot \mathrm{e}^{-\frac{\Delta G^{\mathcal{S}}(X)}{k_{\mathrm{B}}T}}\right]. \tag{10}$$

We then apply Lemma 25 since these $\Omega^{\mathcal{S}}_{s'}$ sets, for different s' are disjoint by Lemma 24, then we can replace the double summation by only one over $\bigcup_{s' \prec s} \Omega^{\mathcal{S}}_{s'}$ and we get

$$Z^{\mathcal{S}}_s = e^{-\frac{\Delta G(d^{\mathrm{M}}(s)) + \Delta G^{\mathrm{assoc}}}{k_{\mathrm{B}}T}} + \sum_{X \in \bigcup_{s' \prec s} \Omega^{\mathcal{S}}_{s'}} \left[e^{-\frac{\Delta G(d^{\mathrm{M}}(s)) + \Delta G^{\mathrm{assoc}}}{k_{\mathrm{B}}T}} \cdot Z_{(s',s)} \cdot e^{-\frac{\Delta G^{\mathcal{S}}(X)}{k_{\mathrm{B}}T}}\right]. \tag{11}$$

We can rewrite Equation (11) as follows:

$$Z^{\mathcal{S}}_s = \sum_{X \in \left[\{\epsilon\} \cup \bigcup_{s' \prec s} \Omega^{\mathcal{S}}_{s'}\right]} \left[e^{-\frac{\Delta G(d^{\mathrm{M}}(s)) + \Delta G^{\mathrm{assoc}}}{k_{\mathrm{B}}T}} \cdot Z_{(s',s)} \cdot e^{-\frac{\Delta G^{\mathcal{S}}(X)}{k_{\mathrm{B}}T}}\right]. \tag{12}$$

The latter transition is justified because if $X = \epsilon$, then $Z_{(s',s)} = 1$ and $\Delta G^{\mathcal{S}}(X) = 0$ since there are no computation strand s' to interact with s as X is the empty configuration.

Let $\Omega_1 = [\{\epsilon\} \cup \bigcup\limits_{s' \prec s} \Omega^{\mathcal{S}}_{s'}]$, $\Omega_2 = \Omega^{\mathcal{S}}_s$, $f : \Omega_1 \to \mathbb{R}$, $f' : \Omega_1 \to \mathbb{R}$, $g : \Omega_2 \to \mathbb{R}$ and $g' : \Omega_2 \to \mathbb{R}$. f', f, g' and g are defined as follows:

- $f'(X) = \Delta G^{\mathcal{S}}(X) + \Delta G(d^{\mathrm{M}}(s)) + \Delta G(d^{\mathrm{R}}(s_l), d^{\mathrm{L}}(s)) + \Delta G^{\mathrm{assoc}}$,
- $f(X) = e^{-f'(X)}$,
- $g'(X') = \Delta G^{\mathcal{S}}(X')$, and
- $g(X') = e^{-g'(X')}$,

where X is a configuration of length l and $s_l \in X$ is the final (right-most) strand on the scaffold of configuration X (which may be at some domain index $\leq N$). We know from Observation 27 that there is a direct one-to-one correspondence, $\alpha : \Omega_1 \to \Omega_2$, defined by $\alpha(X) = s \circledast X$. Notice that function f' is defined in a way that computes the free energy, Equation (1), of the configuration that results from $X' = \alpha(X)$, so $f'(X) = g'(\alpha(X)) = g'(X')$. It directly follows that $f(X) = e^{-f'(X)} = e^{-g'(X')} = g(X')$, then we can apply Lemma 26 directly on Equation (12) using the previous setup for that lemma and we will get $Z^{\mathcal{S}}_s = \sum\limits_{X \in \Omega^{\mathcal{S}}_s} e^{-\Delta G^{\mathcal{S}}(X)/k_{\mathrm{B}}T}$, and this completes the proof. ◀

▶ **Lemma 23.** *For any 1D SDC $\mathcal{S}$, $Z^{\mathcal{S}} = Q^{\mathcal{S}}$, where $Z^{\mathcal{S}}$ is defined in Equation* (6) *and $Q^{\mathcal{S}}$ is the partition function of $\mathcal{S}$ (Equation* (3)*).*

Proof. Since $Z^{\mathcal{S}} = 1 + \sum\limits_{s \in T} Z^{\mathcal{S}}_s$, and from Lemma 22, we know that $Z^{\mathcal{S}}_s = \sum\limits_{X \in \Omega^{\mathcal{S}}_s} e^{-\frac{\Delta G^{\mathcal{S}}(X)}{k_{\mathrm{B}}T}}$. Then, we get the following: $Z^{\mathcal{S}} = 1 + \sum_{s \in T} \sum_{X \in \Omega^{\mathcal{S}}_s} e^{-\frac{\Delta G^{\mathcal{S}}(X)}{k_{\mathrm{B}}T}}$. Again we can apply Lemma 25 directly because these $\Omega^{\mathcal{S}}_s$ sets, for different s, are disjoint from Lemma 24, then we can replace the double summation by only one over $\bigcup_{s \in T} \Omega^{\mathcal{S}}_s$, and with the aid of Observation 28, we get the following:

$$Z^{\mathcal{S}} = 1 + \sum_{X \in \bigcup\limits_{s \in T} \Omega^{\mathcal{S}}_s} e^{-\frac{\Delta G^{\mathcal{S}}(X)}{k_{\mathrm{B}}T}} = 1 + \sum_{X \in \Omega^{\mathcal{S}} \setminus \{\epsilon\}} e^{-\frac{\Delta G^{\mathcal{S}}(X)}{k_{\mathrm{B}}T}} = \sum_{X \in \Omega^{\mathcal{S}}} e^{-\frac{\Delta G^{\mathcal{S}}(X)}{k_{\mathrm{B}}T}}.$$ ◀

5 Kinetics: A simple domain-based kinetic model of 1D SDC

In this section, we first propose a simple kinetic model for the 1D SDC. Second, we implement a simulator for this model (in java and python languages, as a Gillispie simulation [13]) and use it to evaluate existing systems. Third, we use the model to evaluate two hypotheses for improving the kinetics of *constant temperature* 1D SDC systems.

Model. Our kinetic model for a 1D SDC, $\Theta = (S_N, T)$, is a continuous-time Markov chain (CTMC) that satisfies detailed balance [23]. The initial system state as shown in Figure 4(c) is the empty configuration, ϵ, then the simplest version of our kinetic model assumes that the first N steps are merely the binding of N computation strands to the scaffold (no displacement). This assumption is reasonable since we know empirically that hybridization reaction rates are much faster than reactions utilising strand displacement (this assumption can be removed, but we leave it for simplicity). Then any next step will be a sub-step of either toehold exchange or strand displacement, based on a 3-step and 2-step model, respectively described in the following equations and shown in Figure 4(a):

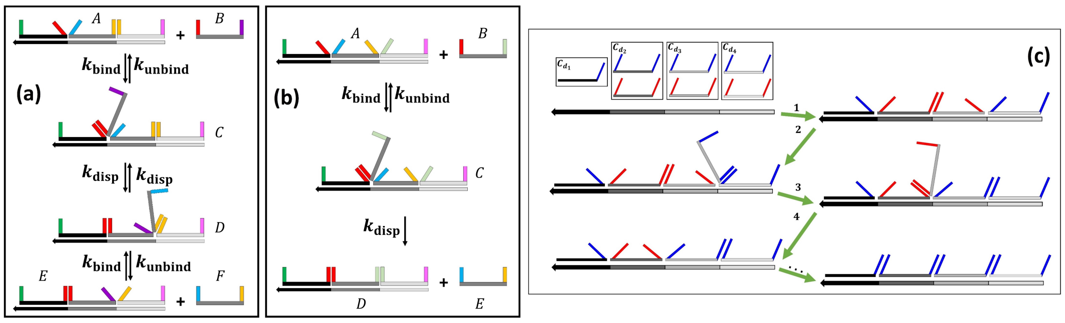


Figure 4 Summary of kinetic 1D SDC model that extends the thermodynamic model in Section 2.1 (and Figure 2). (a) 3 step bind-displace-unbind. (b) 2 step bind-displace. (c) An example computation. We start with the all-empty scaffold, after some fast hybridization events, we have a random scaffold configuration whereupon a sequence of strand displacement events eventually yield the unique thermodynamically-favoured target configuration.

$$\mathrm{A} + \mathrm{B} \underset{\mathrm{k_{unbind}}}{\overset{\mathrm{k_{bind}}}{\rightleftharpoons}} \mathrm{C} \underset{\mathrm{k_{disp}}}{\overset{\mathrm{k_{disp}}}{\rightleftharpoons}} \mathrm{D} \underset{\mathrm{k_{bind}}}{\overset{\mathrm{k_{unbind}}}{\rightleftharpoons}} \mathrm{E} + \mathrm{F}$$

$$\mathrm{A} + \mathrm{B} \underset{\mathrm{k_{unbind}}}{\overset{\mathrm{k_{bind}}}{\rightleftharpoons}} \mathrm{C} \xrightarrow{\mathrm{k_{disp}}} \mathrm{D} + \mathrm{E}$$

where A, B, C, D, E, F are complex names, and the binding and unbinding rates, r_{bind} and r_{unbind} respectively, are $r_{\text{bind}} = [s] {*} k_{\text{bind}}$ and $r_{\text{unbind}} = k_{\text{unbind}} {*} e^{(\Delta G(d^{\mathrm{R}}(s'), d^{\mathrm{L}}(s)) + \Delta G^{\text{assoc}})/k_{\mathrm{B}}T}$, where s, s' are computation strands with matching computation domains, $[s]$ denotes the concentration of s and k_{disp}, k_{bind} and k_{unbind} denote the (single, global) displacement, binding, and unbinding rate, respectively. We used the peppercorn enumerator [2] and NUPACK [7] to obtain these rates.

Simulator. Our simulator for this model works as follows: If we assume the system is currently in state x, the rate for each possible next state x' will be computed, the sum of these rates will be used as a normalization factor to compute a probability for each such next state. Then we compute cumulative density function (CDF) of the random variable that is the next state, and uniformly generate a random number from the interval $[0, 1]$ and use it to determine which next state, x', the system will go to. This process will be repeated forever until the system goes to a halt state, if there is one (which is a state with no next states).

Motivation for two proposals. An interesting challenge for 1D SDC goes thus: suppose we are in the typical situation where our 1D SDC program has a single computation strand, d_1, competing at the first scaffold domain and several computation strand competing at the rest of scaffold domains, and that we begin in an otherwise random configuration. On a large scaffold, there will be many mismatching computational domains, implying many competing random walks (moving to the left and right), and we will in the worst case need to wait something like quadratic time in scaffold length for the single strand at d_1 to "win" all competitions leading to the most enthalpically favoured state. The question we ask here: are there any tricks that we can use to speed up the kinetics?

Note that in this situation the input computation strand, i.e. the leftmost computation strand, plays a crucial role both in determining the target configuration, and in driving the system to that target. Hence, our intuition is that there is value in considering kinetic tricks that push (speed-up) computation in the left-to-right direction. Below, we propose two tricks that build on this intuition.

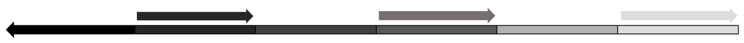


Figure 5 Proposal 1: Scaffold with covers (here at the second, fourth, and sixth scaffold domains).

Proposal 1: Covers. A *cover* is simply a strand consisting of a single domain $\bar{d}$ which is complementary to some scaffold domain d, such that when the cover is at high concentration relative to computation strands that compete at domain d, the cover out-competes the computation strand. If we have covers for every scaffold domain, the scaffold gets coated, or mostly coated, in covers as shown in Figure 5. The key idea here is that covers have no computation domains, and hence do not send left or right randomly-walking signals, intuitively making covers faster to displace. We carried out a number of simulations with various cover excesses. Figure 7 shows the results (scaffold is at 0.1x, computation strands at 1x, and covers at higher excesses).

Proposal 2: Monotonically increasing computation strand concentrations. The intuition here is that by having higher concentrations of computation strands at higher scaffold domain indices (i.e. as we move further from the start position we increase computation strand concentration) the random walk should be biased to move to the right at a higher rate, than with all equal concentrations. The idea is illustrated in Figure 6: First imagine that we have a Bit-Copy system, such that at each scaffold position i we have two computation strands s_i^0, s_i^1 (copying bit 0 or 1, respectively). We set concentrations to be strictly increasing with respect to i, in other words $[s_i^0] = [s_i^1] \leq [s_{i+1}^0] = [s_{i+1}^1]$. Thus, for any choice of bit $b \in \{0, 1\}$, the strand s_{i+1}^b has an equal or higher binding rate than s_i^b. Figure 7 shows result with scaffold at 0.1x, and where a computation strand at scaffold position i has concentration i x.

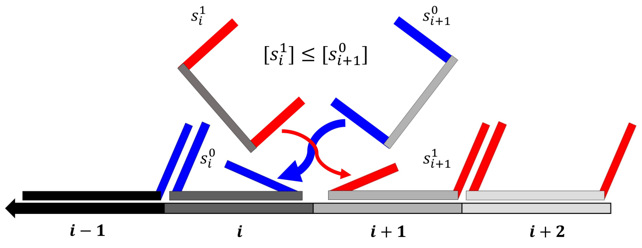


Figure 6 Proposal 2: increasing concentration from left to right. Computation strands further to the right (higher subscript) have higher concentrations than those to the left.

References

1 Tatsuya Akutsu. Dynamic programming algorithms for RNA secondary structure prediction with pseudoknots. *Discrete Applied Mathematics*, 104(1-3):45–62, 2000.

2 Stefan Badelt, Casey Grun, Karthik V Sarma, Brian Wolfe, Seung Woo Shin, and Erik Winfree. A domain-level DNA strand displacement reaction enumerator allowing arbitrary non-pseudoknotted secondary structures. *Journal of the Royal Society Interface*, 17(167):20190866, 2020.

3 Harry M. T. Choi, Maayan Schwarzkopf, Mark E. Fornace, Aneesh Acharya, Georgios Artavanis, Johannes Stegmaier, Alexandre Cunha, and Niles A. Pierce. Third-generation in situ hybridization chain reaction: multiplexed, quantitative, sensitive, versatile, robust. *Development*, 145(12):dev165753, June 2018.

4 Anne Condon, Monir Hajiaghayi, and Chris Thachuk. Predicting minimum free energy structures of multi-stranded nucleic acid complexes is APX-hard. In Matthew Lakin and Petr

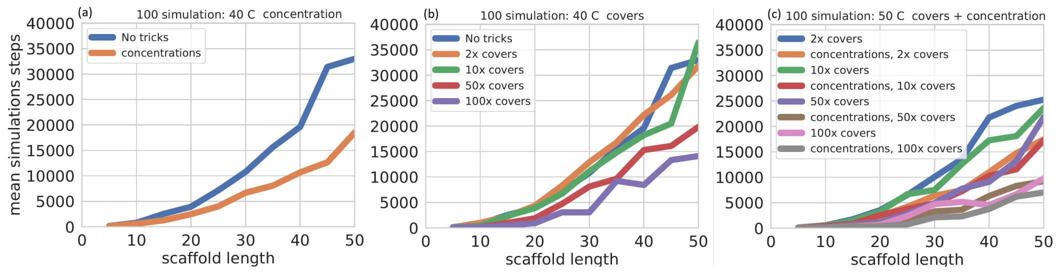


Figure 7 1D SDC kinetics simulations for the Bit-Copy program. 100 simulations were run for each scaffold length that is a multiple of 5 (from 5 to 50), until the target (output) configuration was reached, with the mean number of simulation steps for those 100 shown. (a) The blue curve shows the basic kinetic model (scaffold at 0.1x, computation strands at 1x). Orange shows results for Proposal 2 where scaffold position i has computation strand concentration ix, showing some speed-up. (b) Proposal 1: Covers. Simulations show that high cover concentrations lead to significant speed-ups. (c) Mixing both proposals leads to greater speedups than either.

Šulc, editors, *27th International Conference on DNA Computing and Molecular Programming (DNA 27)*, volume 205 of *Leibniz International Proceedings in Informatics (LIPIcs)*, Dagstuhl, Germany, 2021. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

5 Matthew Cook, Tristan Stérin, and Damien Woods. Small tile sets that compute while solving mazes. In Matthew Lakin and Petr Šulc, editors, *27th International Conference on DNA Computing and Molecular Programming (DNA 27)*, volume 205 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 1–20, Dagstuhl, Germany, 2021. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. Arxiv preprint: `arXiv:2106.12341`.

6 Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT Press, 2022.

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 David Doty, Trent A Rogers, David Soloveichik, Chris Thachuk, and Damien Woods. Thermodynamic binding networks. In *DNA23: The 23rd International Conference on DNA Computing and Molecular Programming*, volume 10467 of *LNCS*, pages 249–266. Springer, 2017.

9 Constantine G Evans. Rgrow tile self-assembly simulator, May 2023. `doi:10.5281/zenodo.7915489`.

10 Constantine G Evans, Jackson O'Brien, Erik Winfree, and Arvind Murugan. Pattern recognition in the nucleation kinetics of non-equilibrium self-assembly. *arXiv preprint arXiv:2207.06399*, 2022.

11 Constantine G Evans and Erik Winfree. Physical principles for DNA tile self-assembly. *Chemical Society Reviews*, 46(12):3808–3829, 2017.

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

13 Daniel T Gillespie. Exact stochastic simulation of coupled chemical reactions. *The journal of physical chemistry*, 81(25):2340–2361, 1977.

14 Raymond Greenlaw, H James Hoover, and Walter L Ruzzo. *Limits to parallel computation: P-completeness theory*. Oxford University Press, USA, 1995.

15 Rune B Lyngsø and Christian NS Pedersen. RNA pseudoknot prediction in energy-based models. *Journal of computational biology*, 7(3-4):409–427, 2000.

16 Cristopher Moore and Stephan Mertens. *The Nature of Computation*. Oxford University Press, 2011.

17 Ruth Nussinov, George Pieczenik, Jerrold R Griggs, and Daniel J Kleitman. Algorithms for loop matchings. *SIAM Journal on Applied mathematics*, 35(1):68–82, 1978.

18 Christos H Papadimitriou. *Computational complexity.* Addison-Wesley, 1994.

19 Andrew Phillips and Luca Cardelli. A programming language for composable DNA circuits. *Journal of the Royal Society Interface*, 6(suppl_4):S419–S436, 2009.

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

21 Paul WK Rothemund. Folding DNA to create nanoscale shapes and patterns. *Nature*, 440(7082):297–302, 2006.

22 John SantaLucia Jr and Donald Hicks. The thermodynamics of DNA structural motifs. *Annu. Rev. Biophys. Biomol. Struct.*, 33:415–440, 2004.

23 Joseph Malcolm Schaeffer, Chris Thachuk, and Erik Winfree. Stochastic simulation of the kinetics of multiple interacting nucleic acid strands. In *DNA Computing and Molecular Programming: 21st International Conference, DNA 21, Boston and Cambridge, MA, USA, August 17-21, 2015. Proceedings 21*, pages 194–211. Springer, 2015.

24 Tristan Stérin, Abeer Eshra, and Damien Woods. Thermodynamically favoured algorithms on a scaffolded DNA computer. In preparation. (Preliminary version presented at The 28th International Conference on DNA Computing and Molecular Programming (DNA28), Track B, 2022.).

25 Petr Šulc, Flavio Romano, Thomas E Ouldridge, Lorenzo Rovigatti, Jonathan PK Doye, and Ard A Louis. Sequence-dependent thermodynamics of a coarse-grained DNA model. *The Journal of chemical physics*, 137(13):135101, 2012.

26 Boya Wang, Cameron Chalk, and David Soloveichik. SIMD||DNA: Single instruction, multiple data computation with DNA strand displacement cascades. In *DNA Computing and Molecular Programming: 25th International Conference, DNA 25, Seattle, WA, USA, August 5–9, 2019, Proceedings 25*, pages 219–235. Springer, 2019.

27 Boya Wang, Siyuan Stella Wang, Cameron Chalk, Andrew Ellington, and David Soloveichik. Parallel molecular computation on digital data stored in DNA. *bioRxiv*, pages 2022–08, 2022.

28 Michael S Waterman and Temple F Smith. RNA secondary structure: A complete mathematical analysis. *Mathematical Biosciences*, 42(3-4):257–266, 1978.

29 Bryan Wei, Mingjie Dai, and Peng Yin. Complex shapes self-assembled from single-stranded DNA tiles. *Nature*, 485(7400):623–626, 2012.

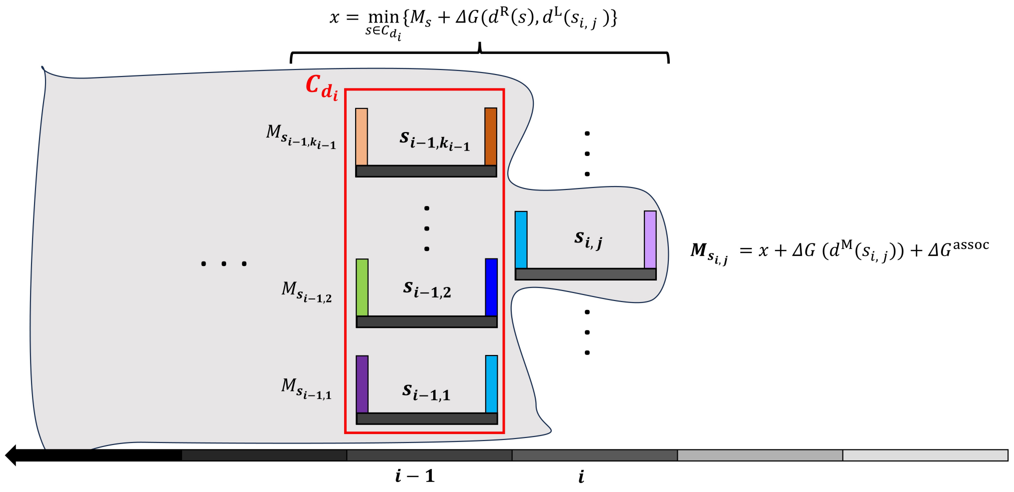
30 Erik Winfree, Rebecca Schulman, and Constantine Evans. The xgrow simulator, 2003.

31 Damien Woods, David Doty, Cameron Myhrvold, Joy Hui, Felix Zhou, Peng Yin, and Erik Winfree. Diverse and robust molecular algorithms using reprogrammable DNA self-assembly. *Nature*, 567(7748):366–372, 2019.

32 Sedigheh Zolaktaf, Frits Dannenberg, Erik Winfree, Alexandre Bouchard-Côté, Mark Schmidt, and Anne Condon. Efficient parameter estimation for DNA kinetics modeled as continuous-time markov chains. In *DNA Computing and Molecular Programming: 25th International Conference, DNA 25, Seattle, WA, USA, August 5–9, 2019, Proceedings 25*, pages 80–99. Springer, 2019.

33 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.

A Appendix: MFE algorithm illustration



$x = \min_{s \in C_{d_i}} \{M_s + \Delta G(d^R(s), d^L(s_{i,j}))\}$
C_{d_i}
$M_{s_{i-1,k_{i-1}}}$
$s_{i-1,k_{i-1}}$
$s_{i-1,2}$
$M_{s_{i-1,2}}$
$s_{i-1,1}$
$M_{s_{i-1,1}}$
$s_{i,j}$
$M_{s_{i,j}} = x + \Delta G(d^M(s_{i,j})) + \Delta G^{\text{assoc}}$
$i-1$
i

Figure 8 Graphical illustration of Algorithm 1 (and for the proof of Theorem 1). The algorithm iterates through scaffold domains, with i being the current scaffold domain. The strand $s_{i,j}$ is the j^{th} strand (out of k_i strands) at scaffold domain i. Strand $s_{i,j}$ has an associated "partial" MFE denoted $M_{s_{i,j}}$ which is the MFE of all configurations that have strand $s_{i,j}$ as their rightmost strand (and have no strands binding to scaffold domains $d_{>i}$). Algorithm 1 proceeds, for each of the $\leq k_i$ strands s at scaffold position i, by populating the list M_{curr}, the j^{th} entry of which will contain the computed value $M_{s_{i,j}}$. In order to compute $M_{s_{i,j}}$, Algorithm 1 makes use of a list M_{prev} that maintains a similar list for index $i-1$, and merely needs to take a min over entries of M_{prev} added to the interaction with computation strand $s_{i,j}$ (shown as the equation for x at the top).

B Appendix: Partition function algorithm illustration

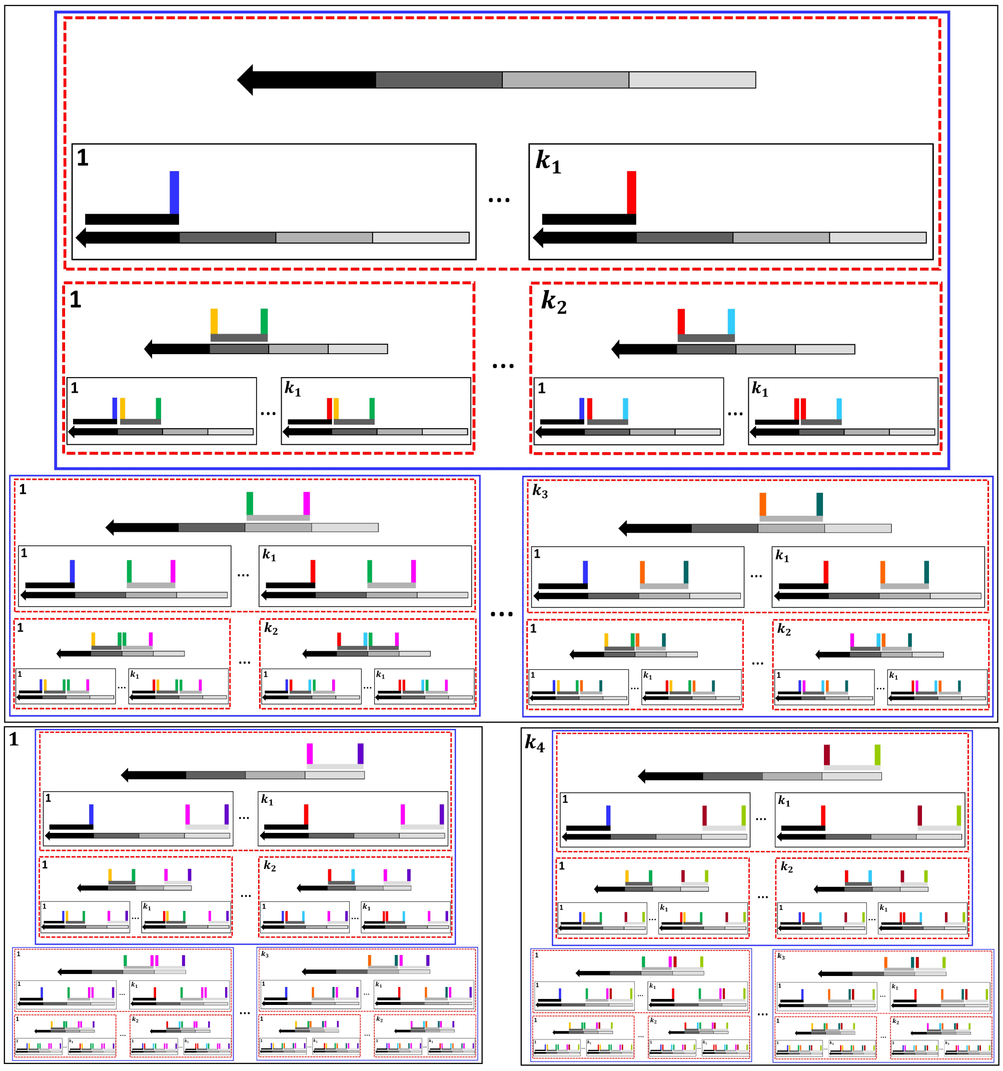


Figure 9 Graphical illustration for Algorithm 2 (and Theorem 2). The coloured boxes contain sets of configurations. The recursive structure in the figure corresponds to the recursive structure of Equation (7): for example, the top blue box corresponds to the summation of the Ω_s-associated partition function for the following partitions: ϵ (empty scaffold on top), $\Omega^{\mathcal{S}}_{s_1^1}, \Omega^{\mathcal{S}}_{s_1^2}, \ldots, \Omega^{\mathcal{S}}_{s_1^{k_1}}$ (top red box), $\Omega^{\mathcal{S}}_{s_2^1}, \Omega^{\mathcal{S}}_{s_2^2}, \ldots, \Omega^{\mathcal{S}}_{s_2^{k_2}}$ (lower level k_2 red boxes in top blue box). Recursively, the computation represented by the top blue box, is then used to compute the partition function for all strands s that compete at domain d_3, and so on.

C Thermodynamics: Basic lemmas

Here, we give some lemmas, proofs are straightforward and hence omitted. Lemma 24 gives a simple method to partition the 1D SDC ensemble $\Omega^{\mathcal{S}}$ using the identity of the rightmost (final) strand (the proof is immediate from the two strands s_1, s_2 being distinct, giving the required partition). The proof of Lemma 25 follows from the associative property of addition and the disjointedness of the involved collection of configuration sets, and Lemma 26 follows from existence of the one-to-one mapping α and how it is defined.

▶ **Lemma 24.** *In any 1D SDC $\mathcal{S}$, for any two distinct computation strands $s_1 \neq s_2$, $\Omega^{\mathcal{S}}_{s_1} \bigcap \Omega^{\mathcal{S}}_{s_2} = \emptyset$.*

▶ **Lemma 25.** *In any 1D SDC $\mathcal{S}$, for any finite collection of disjoint sets of configurations $\Omega_1, \Omega_2, ..., \Omega_n$ and any real-valued function $f : \Omega^{\mathcal{S}} \to \mathbb{R}$,*

$$\sum_{i=1}^{n} \sum_{X \in \Omega_i} f(X) = \sum_{X \in \Omega'} f(X) \text{ where } \Omega' = \bigcup_{i=1}^{n} \Omega_i$$

▶ **Lemma 26.** *In any 1D SDC $\mathcal{S}$, for any two finite sets of configurations Ω_1, Ω_2 and for any two real-valued functions $f : \Omega_1 \to \mathbb{R}$ and $g : \Omega_2 \to \mathbb{R}$ such that there is a one-to-one correspondence, α, between Ω_1 and Ω_2, and $f(X) = g(\alpha(X))$, then $\sum\limits_{X \in \Omega_1} f(X) = \sum\limits_{X' \in \Omega_2} g(X')$.*

▶ **Observation 27.** *As a direct consequence from Lemma 13, for any 1D SDC $\mathcal{S}$, and for any computation strand s, we define a one-to-one correspondence $\alpha : \{\epsilon\} \bigcup_{s' \prec s} \Omega^{\mathcal{S}}_{s'} \to \Omega^{\mathcal{S}}_{s}$, by interlacing the strand s to each configuration X that belongs to the domain of α, $\alpha(X) = X \circledast s$.*

▶ **Observation 28.** *As an important special case of Lemma 13, for any 1D SDC $\mathcal{S}$ of length N, we can show the following $\Omega^{\mathcal{S}}$ partitioning: $\Omega^{\mathcal{S}} = \left[\bigcup_{s \in T} \Omega^{\mathcal{S}}_{s}\right] \bigcup \{\epsilon\}$. By ordering computation strands in T based on their domain indices, we can view $\Omega^{\mathcal{S}}$ as union of a hierarchy of partitions beginning with $\{\epsilon\}$ then all $\Omega^{\mathcal{S}}_{s}$ for all $s \in C_{d_1}$ and so on until all $\Omega^{\mathcal{S}}_{s}$ for all $s \in C_{d_N}$.*

D Computational complexity of MFE

We restate and prove the following theorem. Note that MFE values are ≤ 0.

▶ **Theorem 3.** Given a 1D SDC $\mathcal{S}$ of length $N \in \mathbb{N}$, with $k = O(1)$ computing strands per scaffold domain and bounded domain energies ($\Delta G(d) \in O(1)$, for a domain d binding to its complement), and a finite-precision decimal r, the problem of deciding whether the domain-level MFE of $\mathcal{S}$ is $\leq r$ is in the complexity class L.

Proof. We assume familiarity with deterministic logarithmic space Turing machines and the class L [16, 18]. Let $n = \max(N, \text{len}(r))$ where $\text{len}(r)$ is the number of decimal symbols used to specify r, ignoring leading and trailing 0s. First, we argue that all variables in Algorithm 1 fit in $O(\log n)$ workspace: The MFE value $M^{\mathcal{S}}$ produced by the algorithm is a negative number bounded below by $M \overset{\text{def}}{=} N \cdot (\Delta G(d^M(s)) + \Delta G^{\text{assoc}}) + (N-1) \cdot \Delta G(d^{\text{R}}(s'))$, where s is the strand with the minimum ΔG (strongest) middle domain d^M, and s' is the strand with the minimum ΔG (strongest) right domain d^{R} (i.e. we are taking a min over all domains of the system to compute a putative free energy that no configuration can be less than). Hence $M \in O(N) = O(n)$, and thus $M \leq M^{\mathcal{S}}$ can be written down using $O(\log n)$ bits, keeping in mind that these quantities are all ≤ 0. M_{curr} and M_{prev} in the algorithm are lists of length $k \in O(1)$, and no value stored in these lists is less than M (since M is a lowerbound for the MFE). Finally, the various operations in the algorithm (minimum of a list, addition, assignment, and iteration) are all logspace computable. Hence our MFE problem is in L. ◀

E Application of our partition function algorithm

Armed with a fast partition function algorithm (Algorithm 2) we can analyse 1D SDC systems. Figure 2(d) shows an example result, where we used our partition function algorithm to "simulate" an anneal of the BIT-COPY system, for scaffold lengths up to 400 (i.e. 801 strands), by computing the probability of the target configuration (correct output) at every 2° C from 100° C down to 20° C.[5] We used temperature-dependent domain binding energies computed from DNA sequences from ongoing experimental work [24] (mean ΔG of 24-base scaffold and 12-base computational domains being -20.7 and -11.2 kcal/mol for intended binding, respectively, at 55° C).

For non-complementary (orthogonal) computational domains at adjacent positions on the scaffold, we allowed some binding free-energy (16% of intended binding, an approximation derived from the NUPACK-computed partition function free-energy for a designed sequence set [24]), and a positive (unfavourable) penalty for two bound computational domains (to approximate geometric strain of three helices in close proximity/loop closure). With these minor tweaks, the probability of the target (correct BIT-COPY output) with respect to temperature showed qualitative agreement with experimental results on an $N = 4$ 1D SDC.

Our partition function algorithm allows for analysis of much longer scaffold lengths, useful for future scaled-up designs, with Figure 2(d) showing scaffold lengths up to 400 (i.e. BIT-COPY with 801 strands). This test was meant as a speed test for our algorithm, and not a quantitative prediction of the experimental system, which we leave to future work. In particular there remains some work to calibrate our domain-level model to experiments. For example, lowering the sequence orthogonality constraint from 16% to 8% increases the predicted melting temperature by ≤ 2 degrees for scaffold length ≤ 100, implying that there is some, albeit low, sensitivity to crucial, and difficult to measure, system parameters.

F Source code

Source code for our algorithms can be found at `https://dna.hamilton.ie/software.html`.

[5] An earlier, brute-force exponential-time algorithm could not go beyond scaffold lengths ≥ 15 on our hardware). Note that the target (output) is the structure with MFE.