Geometric Constraint Optimization for Localized Strand Displacement Reactions

Matthew R. Lakin[1]
Department of Computer Science, University of New Mexico, Albuquerque, NM, USA
Department of Chemical & Biological Engineering, University of New Mexico, Albuquerque, NM, USA

Abstract

Localized molecular circuits offer practical advantages over those implemented using components freely diffusing in bulk solution, such as computation speed and component reuse. A common framework for implementing such circuits uses DNA strand displacement reactions with components localized via tethering to a DNA origami tile. While a number of papers have demonstrated the capability of such circuits, design tools for enumerating localized reactions and analyzing their behavior are relatively scarce. The key difficulty in modeling such circuits is that the geometric constraints imposed by the tethering of specific components at specific points on the tile surface are critical in determining whether or not a particular reaction may occur. In previous work, we deployed simple techniques based on random sampling of the structure space in an attempt to find geometric structures for candidate reaction products that satisfy all of the geometric constraints. In this paper, we show that this approach can be enhanced by using an optimization algorithm that takes initial guessed structures that fail to satisfy certain constraints and attempts to refine them into structures that do satisfy all of the constraints. We illustrate this approach on simple example reactions as well as a strand displacement-based signal transmission example from the literature. This work thus advances the state of the art in modeling tools for localized molecular circuits.

2012 ACM Subject Classification Computer systems organization → Molecular computing

Keywords and phrases Localized circuits, reaction enumeration, DNA strand displacement, constraint solving, geometry, molecular computing

Digital Object Identifier 10.4230/LIPIcs.DNA.32.5

Supplementary Material *Software (Source code)*: `https://github.com/matthewlakin/LocalizedEnumeratorOptimization` [24]
archived at `swh:1:dir:095f551c8228cf74f1548237675d8391b3963452`

Funding This material is based upon work supported by the National Science Foundation under grants 1518861, 1814906, and 2044838.

Acknowledgements The author thanks Abigail Carlson for early work on a preliminary version of the system reported in this paper.

1 Introduction

Molecular computing based on DNA strand displacement has been shown to be a powerful tool for implementing computation at the nanoscale. Networks of components diffusing freely in bulk solution have been used for many years to experimentally implement a range of computational paradigms, including digital logic circuits [38], population protocols [9], and recurrent neural networks [10, 39], including those trainable *in situ* [11].

In addition, several groups have developed variants of strand displacement systems in which some, or all, of the circuit components are localized to the surface of a DNA nanostructure. Typically this means the face of a two-dimensional DNA origami tile [40], where staple extension provides a simple means of programming the positions of individual

[1] Corresponding author

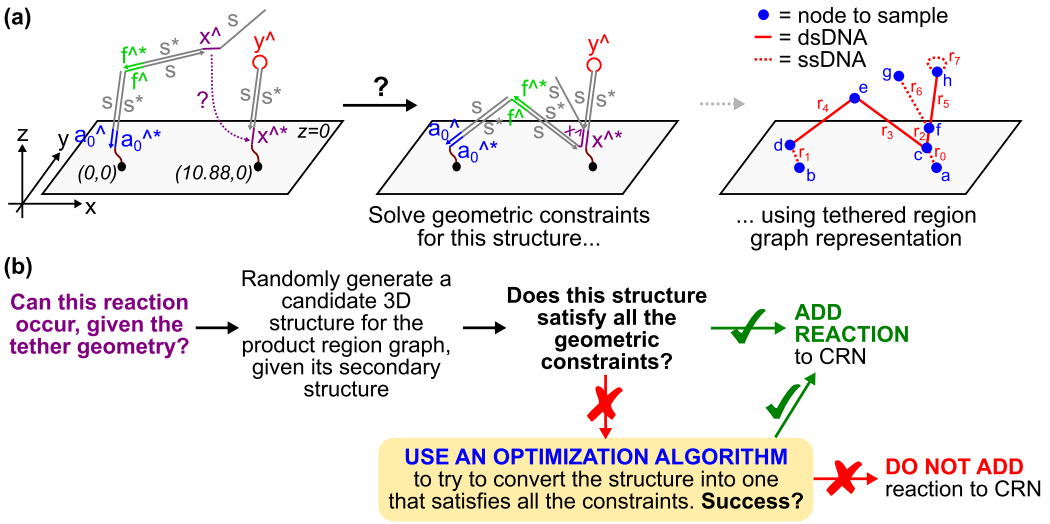

Figure 1 Geometric reaction enumeration for tethered DNA strand displacement networks. **(a)** Our approach uses an algorithm to determine whether a particular localized DNA interaction is geometrically feasible: first the putative product secondary structure is determined, then a candidate three-dimensional physical structure for this product is generated, based on a "region graph" representation that translates the strand graph representation language into one more suited for geometric solving. If a geometrically plausible structure for the region graph can be found, we determine that the reaction may proceed based on geometry. **(b)** In this paper we improve the reliability of this process by taking a structure that initially fails this plausibility test and attempting to convert it into one that *is* plausible, using non-linear optimization routines.

components on the surface. In particular, previous work from the Seelig group [8, 33] and the Reif group [5–7] has demonstrated the construction of multi-step signal processing cascades and logic circuits using hairpin-based strand displacement components. This approach to DNA computing offers potential advantages over the purely solution-phase approach. In particular, localized reactions can proceed more rapidly than solution-phase interactions because the reactants are tethered in close proximity, meaning that there are fewer degrees of freedom for reactant diffusion. Furthermore, nucleotide sequences can be reused between components because their position on the tile surface encodes their position in the circuit, unlike in solution-phase systems where each distinct logic gate must be implemented with sequences that are sufficiently distinct to prevent undesired crosstalk.

The issue of molecular geometry is typically not directly considered in reaction enumeration in solution-phase molecular circuits. The well-mixed assumption means that spatial considerations do not arise, and it is assumed that components may approach each other in whatever orientation is required in order to interact. For example, early versions of the Visual DSD system [28, 31] restricted to a particular class of linear heteropolymer structures that, while expressive, did not allow for the kinds of structure where geometric constraints might stymie reaction progress, such as loops. Those were admitted in later versions of the tool that used a more expressive strand graph syntax [36], but certain geometric considerations were handled via *ad hoc* side conditions on reaction rules, for example, to prevent any binding interaction where one single-stranded domain is enclosed within a hairpin loop. This was not ideal because it failed to capture more subtle interactions of geometry with molecular design.

The collection of design tools available for modeling and analyzing the behavior of tethered DNA strand displacement systems is thus somewhat less than developed than that for more standard approaches to molecular computing. Some early work on the Visual DSD system

did introduce a syntax for representing tethered structures attached to DNA origami tiles, along with a manual system for annotating which components attached to the tile could interact with each other [27]. However, that approach is unsatisfactory because it places the burden for solving the key aspect of this reaction enumeration problem onto the user, namely, how to solve the geometric constraints inherent in tethered molecular circuits. In previous work, we tackled this problem using various approaches to automate the solving of the geometric constraint problems inherent in the enumeration of localized DNA strand displacement reaction networks. These constraints often boil down to determining whether two localized components are tethered close enough to each other on the surface to interact, as illustrated in Figure 1(a), though in the case of hairpin-based systems there are also often geometric constraints imposed by sequestering toeholds in a tight hairpin loop. In our earliest work [29], we used the Z3 SMT solver [14] to solve some instances of constraint problems arising in geometric reaction enumerators. However, that approach had performance issues due to the theoretically complete but sometimes extremely inefficient process of solving constraints modulo theories of real arithmetic [21].

We therefore adopted a conceptually simpler approach of repeated, randomized sampling of the structure space [23]. We used this approach to search for geometric structures for putative product species that satisfies the constraints implied by their secondary structures. Using a simple biophysical model, we successfully integrated this approach into a reaction enumerator over strand graphs for non-tethered species [22] and showed that it could successfully solve the geometric constraints involved in hairpin binding and remote toehold DNA strand displacement [19]. Most recently, we applied the same approach to a structure sampling enumerator for tethered DNA strand displacement systems [25]. However, the naïve approach to structure sampling taken in those papers means that it can sometimes take large numbers of attempts to find a structure that satisfies all of the constraints.

In this work, we attempt to mitigate some of the randomness in those approaches by using optimization algorithms to attempt to turn sampled structures that do not satisfy all of the constraints into ones that do. This approach is outlined in Figure 1(b). Our goal is to increase the probability of each attempt finding a satisfying structure, if one exists. We can then try fewer sampling attempts, though each one may do a lot more work. In the sections that follow, we define the key concepts and outline our implementation's approach to optimization-based constraint solving. We then apply the algorithm to a simple tethered hybridization reaction and to both two- and three-stator versions of the hairpin-based signal propagation line of Chatterjee et al [8]. In particular, the three-stator variant is very difficult for the naïve structure sampling approach to enumerate because it involves two loop constraints that must be satisfied simultaneously. The results from our algorithm in the three-stator case also find a number of structures that are geometrically plausible in our biophysical model but which are not accounted for in the idealized chemical reaction networks reported for those systems. This suggests that localized strand displacement systems may be more susceptible to off-target interactions, driven by the proximity and flexibility of the reactants. Our work thus improves existing capabilities for enumeration of reaction networks for localized molecular computing systems and also suggests directions for future research into circuit designs.

2 Tethered geometric reaction enumeration

This paper builds upon our previous geometric reaction enumerators using structure sampling [22] for tethered strand graphs [25]. We recapitulate the key points below, and refer the reader to those papers for further details.

2.1 Tethered strand graphs

A tethered strand graph is a strand graph [36] in which some strands are annotated with a tether at either the 5' or 3' end. We assume that all tethers on a single strand are attached to the same tile; interactions between two tiles are thus prohibited. Each component tethered to a particular tile is represented as a single connected strand graph; we use a higher-level construct to group all tethers attached to a single tile. (Here we assume, as in our previous work [22], that vertices and sites in tethered strand graphs are indexed starting from zero; this contrasts with the original strand graph paper [36] where indexing started from one.) The fundamental definition of a tethered strand graph is presented below, reproduced from [25]:

▶ **Definition 1** (Tethered strand graphs.). *For $V \subseteq \mathbb{N}$ and $length : V \to \mathbb{N}$, we define the following helper functions:*

- $legal_S(V, length) = \{(v, n) \in \mathbb{N}^2 \mid v \in V \wedge n \leq length(v)\}$ *to be the set of legal sites, where a site $s = (v, n)$ is a pair of natural numbers such that v is a vertex (representing a single strand) and n is a position on that vertex (representing a sequence domain).*
- $legal_E(V, length) = \{S \subseteq legal_S(V, length) \mid |S| = 2\}$ *is the set of legal edges, where an edge $e = \{s_1, s_2\}$ is a set containing two sites, such that $s_1 \neq s_2$.*

*A **tethered strand graph** $G = (V, length, color, tether5', tether3', A, toehold, E)$, where:*

- $V = \{1, \ldots, N\}$ *is the set of vertices; each vertex is a natural number.*
- $color : V \to \mathbb{N}$ *is a function assigning a natural number "color" to each vertex.*
- $length : V \to \mathbb{N}$ *is a function assigning a length to each vertex, such that $color(v_1) = color(v_2) \implies length(v_1) = length(v_2)$. That is, the length is really a function of the color.*
- $tether5' : V \to ((\mathbb{N} \times \mathbb{N}) \uplus \{\perp\})$ *and* $tether3' : V \to ((\mathbb{N} \times \mathbb{N}) \uplus \{\perp\})$ *are functions assigning either a pair of (x, y)-coordinates to the corresponding tethered strand terminus of each vertex, or $\perp$ if that strand terminus is not tethered, such that at most one of $tether5'(v)$ and $tether3'(v)$ is not $\perp$, for any v. That is, at most one terminus of each strand can be tethered. Note that vertices of the same color may have different tethers, as the tether is a property of the individual strand not the vertex color.*
- $A \subseteq legal_E(V, length)$ *is a set of admissible edges, such that $color(e_1) = color(e_2) \implies (e_1 \in A \iff e_2 \in A)$ and such that $(legal_S(V, length), A)$ forms a bipartite graph, which ensures that admissible edges connect complementary domains. Furthermore, for an edge $e = \{(v_1, n_1), (v_2, n_2)\}$ we define its color to be: $color(e) = \{(color(v_1), n_1), (color(v_2), n_2)\}$.*
- $toehold : A \to \mathbb{B}$ *is a function that returns true on all admissible edges between toehold domains and false on all admissible edges on long domains, such that $color(a_1) = color(a_2) \implies toehold(a_1) = toehold(a_2)$.*
- $E = \{e_1, \ldots, e_M\} \subseteq A$ *is a set of current edges, such that $(i \neq j) \implies e_i \cap e_j = \varnothing$. That is, at any given instant there can be at most one edge connected to any given site.*

2.2 Geometric reaction enumeration for tethered strand graphs

We now define reaction enumeration over tethered strand graphs. The first component of this definition is a transition relation $G \to G'$ between pairs of tethered strand graphs; this is as defined in our previous work [25] and is reproduced as Figure A.1 for convenience. These rules specify the fundamental kinds of transition that may occur in our system: binding of complementary domains, toehold unbinding, and three- and four-way branch migration. Each rule has a premise of the form $plausible(G')$, which requires that the candidate product structure G' must be shown to be "geometrically plausible" for the transition to be permissible. To define the chemical reaction network (CRN) semantics, we distinguish two kinds of species:

- *free species* S are connected strand graphs that contain no tethers;
- *tile species* $[[\,\overline{S}\,]]$ represent a single DNA origami tile with attached components, where $\overline{S} = S_1, \ldots, S_n$ is a finite collection of distinct, connected strand graphs, each of which contains at least one tether.

Then, on top of the $\rightarrow$ relation between individual tethered strand graphs we build a transition relation $\Rightarrow$ between collections of free or tile species. In addition to unimolecular reactions, we permit bimolecular interactions either between two free species as reactants, or between one free and one tile species. We do not permit interactions between two distinct tiles. Furthermore, uni- or bimolecular interactions between species tethered to the same tile are lifted to unimolecular reactions at the level of the tile species. The rules defining this relation were introduced in our previous work [25]; we reproduce them here as Figure A.2.

2.3 Forming geometric constraint sets from strand graphs

The first step toward forming the set of geometric constraints associated with a strand graph G is to form the associated region graph, $\mathit{regiongraph}(G) = (RGV, RGE)$. The rules defining this process are reproduced from our previous work [22] for convenience, as Appendix B. The region graph is an alternative representation of the structure in which vertices, rather than representing domains, now represent the *boundaries* between domains, at which the structure has flexibility. (Figure 1(a) illustrates the region graph produced for an example tethered structure.) We must provide a three-dimensional (x, y, z) coordinate for each vertex of the region graph in order to test for plausibility of the corresponding structure. The edges of the region graph represent contiguous segments of either single- or double-stranded DNA. In our simplified biophysical model, we assume that double-stranded regions are short enough relative to the persistence length of DNA duplexes to be modeled as rigid rods, whereas single-stranded regions are assumed to infinitely flexible, as are all joints between regions.

We note that this process implicitly implements the "hybridization constraints" that we used in previous work [30], as domains hybridized to each other end up being collapsed into the same vertices of the region graph (which we consider acceptable because we neglect the width of the DNA double helix in our simplified biophysical model). Thus, constraints that remain to be solved may take five different forms:

- **Double-stranded region length constraints:** $\mathsf{eqdistconstr}(N, N', d)$, which specifies that the distance between vertices N and N' must be precisely equal to d;
- **Single-stranded region length constraints:** $\mathsf{maxdistconstr}(N, N', d)$, which specifies that the distance between vertices N and N' must be at most d;
- **Nick angle constraints:** $\mathsf{nickconstr}(N, N', N'', \theta)$, which specifies that the magnitude of the angle of deviation from one of N or N' to the other, when passing through the nick, is less than or equal to θ (helpful when analyzing certain examples);
- **Tether location constraints:** $\mathsf{tetherconstr}(N, (x, y))$, which specifies that the vertex N must be tethered at the coordinates $(x, y, 0)$ on the tile, whose surface is assumed to form the $z = 0$ plane of its local coordinate system; and
- **Tile constraints:** $\mathsf{zcoordconstr}(N)$, which specifies that the z-coordinate of the vertex N must be at least zero, that is, not on the "wrong" side of the tile.

We now assume that the input strand graph G has been converted into the corresponding region graph, $\mathit{regiongraph}(G) = (RGV, RGE)$. To decide which kind of constraint to use for a given edge of a region graph, we write $\mathit{dse}(RGE)$ for the subset of edges from the region graph that correspond to *double-stranded* regions and $\mathit{sse}(RGE)$ for the subset of edges from the region graph that correspond to *single-stranded* regions. We write $\mathit{edgelength}(e)$ for the

length in nanometers associated with the region graph edge e, which corresponds to the *total* length for double-stranded domains and the *maximum* length for single-stranded domains. Then, the constraint set for a given strand graph is as follows.

▶ **Definition 2** (Constraint sets for a given strand graph). *We define the set of constraints, $\mathcal{C}(G)$, that corresponds to the strand graph G, as follows:*

$$\begin{aligned}\mathcal{C}(G) = \{&\mathsf{eqdistconstr}(N, N', d) \mid \{N, N'\} \in \mathit{dse}(\mathit{RGE}) \wedge \mathit{edgelength}(\{N, N'\}) = d\} \cup \\ \{&\mathsf{maxdistconstr}(N, N', d) \mid \{N, N'\} \in \mathit{sse}(\mathit{RGE}) \wedge \mathit{edgelength}(\{N, N'\}) = d\} \cup \\ \{&\mathsf{nickconstr}(N, N', N'', \theta) \mid \{N, N'\} \in \mathit{dse}(\mathit{RGE}) \wedge \{N', N''\} \in \mathit{dse}(\mathit{RGE})\} \cup \\ \{&\mathsf{tetherconstr}(N, (x, y, 0)) \mid ((v, 0), 5') \in N \wedge \mathit{tether5'}(v) = (x, y)\} \cup \\ \{&\mathsf{tetherconstr}(N, (x, y, 0)) \mid ((v, \mathit{length}(v) - 1), 3') \in N \wedge \mathit{tether3'}(v) = (x, y)\} \cup \\ \{&\mathsf{zcoordconstr}(N) \mid N \in \mathit{RGV}\}\end{aligned}$$

2.4 Constraint satisfaction for geometric constraint sets

A **geometric structure** $\mathcal{S}$ for a tethered strand graph G is a mapping from the set RGV of vertices in the corresponding region graph $\mathit{regiongraph}(G)$ to 3D Cartesian coordinates. If $\mathcal{S}$ maps vertex N to coordinate (x, y, z), then we write $\mathcal{S}(N) = (x, y, z)$. A candidate structure $\mathcal{S}$ satisfies a finite set of constraints $\{c_1, \ldots, c_n\}$, which we write as $\mathcal{S} \models \{c_1, \ldots, c_n\}$, if $\mathcal{S}$ simultaneously satisfies each individual constraint c_i in the set. We overload notation and write this as $\mathcal{S} \models c$, where constraint satisfaction will be defined for each of the different kinds of constraint specified above. First, however, a few auxiliary functions: we write $\mathit{dist}(C_1, C_2)$ for the Euclidean distance between the 3D coordinates C_1 and C_2. Also, we write $\mathit{devangle}(\mathcal{S}(C), \mathcal{S}(C'), \mathcal{S}(C''))$ for the magnitude of the angle between the vector $C' - C$ and $C'' - C$, when placed tail-to-tail. We can now define constraint satisfaction.

▶ **Definition 3** (Constraint satisfaction by geometric structures). *We define constraint satisfaction for the different kinds of constraint by cases, as follows.*

- *If* $c \equiv \mathsf{eqdistconstr}(N, N', d)$ *then* $\mathcal{S} \models c \iff \mathit{dist}(\mathcal{S}(N), \mathcal{S}(N')) = d$.
- *If* $c \equiv \mathsf{maxdistconstr}(N, N', d)$ *then* $\mathcal{S} \models c \iff 0 \leq \mathit{dist}(\mathcal{S}(N), \mathcal{S}(N')) \leq d$.
- *If* $c \equiv \mathsf{nickconstr}(N, N', N'', \theta)$ *then* $\mathcal{S} \models c \iff 0 \leq \mathit{devangle}(\mathcal{S}(N), \mathcal{S}(N'), \mathcal{S}(N'')) \leq \theta$.
- *If* $c \equiv \mathsf{tetherconstr}(N, (x, y, 0))$ *then* $\mathcal{S} \models c \iff \mathcal{S}(N) = (x, y, 0)$.
- *If* $c \equiv \mathsf{zcoordconstr}(N)$ *then* $\mathcal{S} \models c \iff \exists x, y, z. \mathcal{S}(N) = (x, y, z) \wedge z \geq 0$.

We say that a tethered strand graph G is *plausible*, which we write as $\mathit{plausible}(G)$, iff there exists a geometric structure $\mathcal{S}$ such that $\mathcal{S} \models \mathcal{C}(G)$ holds. *The key contribution of this paper is improved algorithms for finding a satisfying geometric structure for a given region graph, and hence for testing geometric plausibility.*

3 Enhanced constraint solving over tethered region graphs

3.1 Energy functions over geometric structures

To allow for iterative improvement of an initially unsatisfying geometric structure, we need a metric to define how far away from satisfaction the current geometric structure actually is. Therefore, in this section we define an *energy function* over geometric structures for a given tethered strand graph with respect to its constraint set. We note that this is not an energy in the strict sense, but will merely serve as a loss function to be optimized over below.

▶ **Definition 4** (Energies of geometric structures). *Given a finite set of constraints* $\{c_1, \ldots, c_n\}$ *and a geometric structure* $\mathcal{S}$*, we define the function energy as follows:*

$$energy(\mathcal{S}, \{c_1, \ldots, c_n\}) = \sum_{i \in \{1,\ldots,n\}} energy(\mathcal{S}, c_i)$$

where the energy associated with each individual constraint is defined by cases on the different kinds of constraint introduced above:

- *If* $c \equiv \mathsf{eqdistconstr}(N, N', d)$ *then* $energy(\mathcal{S}, c) = (dist(\mathcal{S}(N), \mathcal{S}(N')) - d)^2$.
- *If* $c \equiv \mathsf{maxdistconstr}(N, N', d)$ *and* $dist(\mathcal{S}(N), \mathcal{S}(N')) \leq d$ *then* $energy(\mathcal{S}, c) = 0$. *Otherwise,* $energy(\mathcal{S}, c) = (dist(\mathcal{S}(N), \mathcal{S}(N')) - d)^2$.
- *If* $c \equiv \mathsf{nickconstr}(N, N', N'', \theta)$ *and* $devangle(\mathcal{S}(N), \mathcal{S}(N'), \mathcal{S}(N'')) \leq \theta$ *then* $energy(\mathcal{S}, c) = 0$. *Otherwise,* $energy(\mathcal{S}, c) = (devangle(\mathcal{S}(N), \mathcal{S}(N'), \mathcal{S}(N'')) - \theta)^2$.
- *If* $c \equiv \mathsf{tetherconstr}(N, (x, y, 0))$ *then* $energy(\mathcal{S}, c) = dist(\mathcal{S}(N), (x, y, 0))^2$.
- *If* $c \equiv \mathsf{zcoordconstr}(N)$ *and* $\mathcal{S}(N) = (x, y, z)$ *and* $z \geq 0$ *then* $energy(\mathcal{S}, c) = 0$. *If* $z < 0$ *then* $energy(\mathcal{S}, c) = 10^{100}$.

▶ **Lemma 5** (Energies are non-negative). *For all finite constraint sets* $\{c_1, \ldots, c_n\}$ *and geometric structures* $\mathcal{S}$*:*

$$energy(\mathcal{S}, \{c_1, \ldots, c_n\}) \geq 0.$$

Proof. This follows straightforwardly from the fact that the total energy is the sum of the energies associated with each individual constraint, $energy(\mathcal{S}, c_i)$. Each of these is either positive or zero, which can be seen by case analysis over constraints as defined above. ◀

▶ **Lemma 6** (Satisfaction iff energy is zero). *For all finite constraint sets* $\{c_1, \ldots, c_n\}$ *and geometric structures* $\mathcal{S}$*:*

$$\mathcal{S} \models \{c_1, \ldots, c_n\} \iff energy(\mathcal{S}, \{c_1, \ldots, c_n\}) = 0.$$

Proof. By Lemma 5, the energy values for individual constraints cannot be negative. Therefore, the only way the energy of the entire constraint set can be zero is when the energy associated with every individual constraint is zero. We then proceed by case analysis on the different kinds of constraint, comparing the relevant cases from the definitions of constraint satisfaction by $\mathcal{S}$ and of the energy function. If a constraint involves an inequality, then the energy is zero in the region satisfying the inequality and the square of the magnitude by which the inequality is violated. For constraints involving an equality, the energy is just the square of the magnitude of the error when comparing the actual result of the calculation to that required to satisfy the constraint. Thus, in every case, the constraint is satisfied precisely when the energy function evaluates to zero, which gives the result. ◀

Given the properties established above, it follows that minimizing such an energy function over a strand graph using a numerical optimization routine thus provides a possible route to convert an unsatisfying geometric structure into a satisfying one.

3.2 Optimizing energies of sampled structures

Given an energy function defined over structures for a given set of geometric constraints, we can consider optimizing these structures in an attempt to minimize the energies to zero which, by Lemma 6, would correspond to a structure that simultaneously satisfies all of the constraints. This raises the question of how to choose the starting structure for the

optimization process. Here we draw on our previous work on structure sampling over region graph structures [22, 23, 25], which yields structures that satisfy most of the constraints by construction, leaving some constraints (those that close a loop, either explicitly in the strand graph or implicitly via the underlying origami tile) to be checked *post hoc*.

We refer the reader to those papers for the details of this process; briefly, sampling begins from a randomly chosen graph of maximum degree (in the case of free species) or a randomly chosen tether (in the case of a species tethered to a tile). An adjacent region is chosen at random to be sampled for its directionality unit vector and, in the case of single-stranded regions, its length. (The length of double-stranded regions is fixed.) Double-stranded regions are always sampled before single-stranded regions, when available, as the single-stranded regions are the ones that can "take up the slack" to enable constraints to be satisfied when a loop is eventually closed. Crucially, this iterative sampling process determines, as an offshoot of its operation, an ordering on the regions that will be critical in our approach below.

In early experiments on this system we generated an initial structure via sampling and, if that structure failed to satisfy all of the constraints, ran an optimization procedure on the (x, y, z)-coordinates of the region graph vertices directly. While this could, in some cases, produce energies very close to zero, the resulting structures never actually satisfied the original constraints. This was because the energy penalty for modifying the length of a (supposedly fixed-length) double-stranded region was often worth paying to minimize the overall energy of the system. We therefore could not sufficiently constrain the lengths of these double-stranded regions so that the resulting structure was geometrically plausible without subsequent adjustment.

This led us to reimagine our optimization algorithm by lifting it to the level of structure sampling operations. Rather than operating on the coordinates of the structure directly, the optimization variables became those that guide the structure sampling process, *assuming a particular ordering on the sampled regions.* To specify this process, here we reformulate the structure sampling process in a slightly more formal manner in terms of *sampling actions.*

▶ **Definition 7** (Sampling actions.)**.** *A* sampling action *A is one of the following:*

- $\mathsf{fixVertex}(N, C)$*, which means that the 3D Cartesian coordinates of region graph vertex N are set to be C. This is used to establish which vertex is at the origin in the case of a free species, and to establish which vertices are tethers in the case of a tile species;*
- $\mathsf{doubleStranded}(N, v, d, N')$*, which means that the coordinates for the unsampled vertex N' are obtained from those from the previously-sampled vertex N by moving along the sampled directionality unit vector v for a distance d equal to the fixed length of the corresponding double-stranded region linking N to N'; and*
- $\mathsf{singleStranded}(N, v, d, d_{max}, N')$*, which means that the coordinates for the unsampled vertex N' are obtained from those from the previously-sampled vertex N by moving along the sampled directionality unit vector v for a distance d sampled from between zero and the maximum length d_{max} of the corresponding single-stranded region linking N to N'.*

Using the structure sampling algorithms from our previous work [25], it follows that a single structure sampling run over a region graph $\mathit{regiongraph}(G) = (RGV, RGE)$ produces a randomly chosen ordered list $\overline{A}$ of sampling actions, such that every vertex $N^* \in RGV$ is sampled precisely once – that is, appears in precisely one sampling action in $\overline{A}$ of the form $\mathsf{fixVertex}(N^*, C)$ for some C, or of the form $\mathsf{doubleStranded}(N, v, d, N^*)$ or $\mathsf{singleStranded}(N, v, d, d_{max}, N^*)$ for some N, v, d, d_{max}.

Given a region graph and a sequence of sampling actions generated from it during a structure sampling attempt, the corresponding geometric structure for that region graph can be reconstructed. Furthermore, from this sequence of sampling actions an ordered sequence

of floating-point parameters can be extracted that is suitable as input to an optimization algorithm. This sequence can be derived by traversing the sequence $\overline{A}$ and appending the appropriate parameters for each sampling action to the sequence in turn, as follows:

- For a sampling action $\mathsf{fixVertex}(N, C)$, no new parameters are added.
- For a sampling action $\mathsf{doubleStranded}(N, v, d, N')$, new parameters $[v_x, v_y, v_z]$, the components of the directionality vector v, are added.
- For a sampling action $\mathsf{singleStranded}(N, v, d, d_{max}, N')$, new parameters $[v_x, v_y, v_z, d]$, the ordered components of the directionality vector v and the sampled length d, are added.

Via a similar mechanism, bounds for these parameters can be extracted: for the components of directionality unit vectors, the lower bounds can be set to -1 and the upper bounds can be set to $+1$. For the sampled lengths of single-stranded regions, the lower bound is 0 and the upper bound is the d_{max} value from the sampling action.

The key advantage of this approach is that, by modifying these parameters within their bounds, our optimization attempts tend to preserve satisfiability of as many of the individual geometric constraints as possible. This is because instead of optimizing the coordinates directly, we optimize the underlying structural parameters from which the coordinates are derived. In particular, the fixed lengths of double-stranded domains are not modified. This allows us to run the optimization routine over an initial failed structure sampling attempt before taking the resulting sequence of modified parameters and, in conjunction with the original sequence of sampling actions, reconstructing the corresponding geometric structure for these optimized parameters. We can then recheck constraint satisfaction to confirm that the new structure is indeed geometrically plausible.

4 Results

We implemented a prototype of the algorithm presented above using the Python programming language. In this section, we present results obtained from testing its capabilities on several example systems. Code for our prototype is available online under an open source license from `https://github.com/matthewlakin/LocalizedEnumeratorOptimization`. In these tests, we used the following biophysical parameters: the length per nucleotide in double-stranded regions was 0.34 nm and the length per nucleotide in single-stranded regions was 0.68 nm. These constant values have been used by other similar studies in the literature [12, 19].

4.1 Simple tethered hybridization example

We began by testing our system on a simple example comprising two tethered strands with complementary domains that may or may not hybridize, depending on the relative positions of the two tethers. This example is illustrated in Figure 2(a), including tethered process calculus-style code and a description of the domain lengths (5nt for the spacer domain spcr and 30nt for the x binding domain). In this example, a single reaction may be possible, namely, the irreversible binding reaction of the complementary domains, x and x^*. This makes it a straightforward initial test of our new constraint solving approach.

As a baseline, we benchmarked our enhanced system against the system from our previous work [25], which relies solely on structure sampling to try to find satisfying structures for a candidate reaction product. In that approach, structure sampling is carried out as outlined above, except that a sampled structure that fails to satisfy the constraints is immediately discarded and another sampling attempt is made, until the maximum number of allowed attempts is reached. Here, we aimed to contrast the sampling-only approach with our optimization-based approach, so we deliberately used smaller numbers of sampling attempts (10, 25, or 50) than in our previous work. In all cases, we tested inter-tether distances

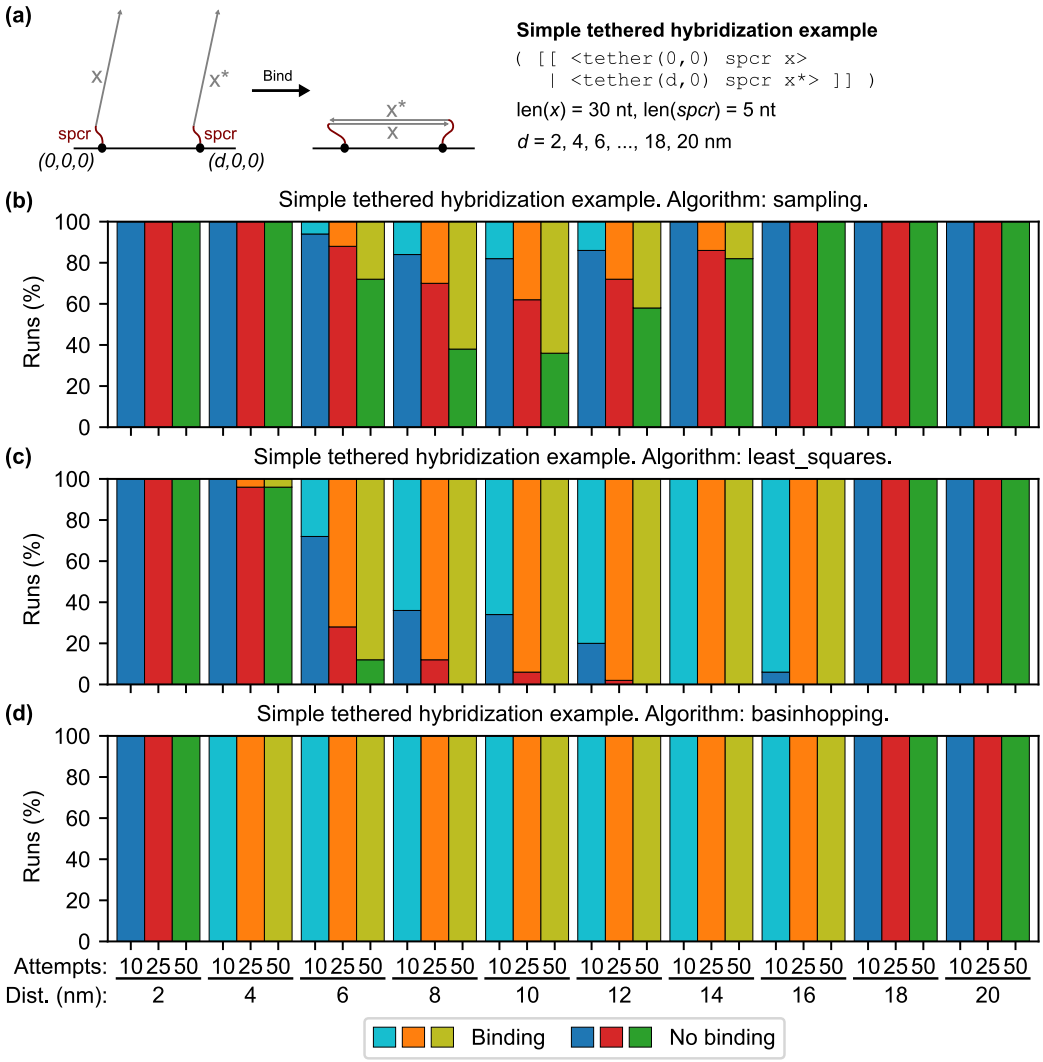

Figure 2 Testing geometric constraint optimizers on a simple tethered hybridization example. **(a)** System outline, with process calculus-style code and other parameters such as domain lengths in nucleotides. **(b)** Summary of enumeration results using the sampling-only approach. For each inter-tether distance and number of attempts, 50 runs were carried out. In each case, the percentages of runs in which the binding reaction was or was not enumerated are plotted as stacked bars. **(c)** Similar summary of enumeration results using structure optimization using the `least_squares` method. **(d)** Similar summary of enumeration results using structure optimization using the `basinhopping` method. This approach proved the most reliable, with a 100% success rate.

between 2 nm and 20 nm in increments of 2 nm. We generated 100 distinct pseudo-random number generator seeds for each test and computed statistics from these runs to study the reliability of the results.

Perhaps unsurprisingly, with so few sampling attempts permitted the sampling-only system failed to reliably enumerate the binding reaction: only toward the middle of the tested region (8 and 10 nm inter-tether distances) did it achieve greater than 50% success (Figure 2(b)). Given the maximum lengths of the single-stranded spacers, satisfying structures should exist for inter-tether distances between 3.4 and 17 nm.

We next compared this to an optimization-based reaction enumerator using the approach outlined above. In our first tests, we used an optimization algorithm based on a non-linear least-squares minimizer: the `least_squares` function from the `scipy.optimize` Python library [45], using the "Trust Region Reflective" algorithm. This approach can directly handle problems with constraints on the variables, and we used this capability to constrain the vectors and single-stranded region lengths as outlined above. The results using this approach (Figure 2(c)) were much more reliable than those from the sampling-only approach, showing 100% (or close to 100%) enumeration of the binding reaction across much of the parameter space where we would expect solutions to exist. However, particularly for lower numbers of attempts and shorter inter-tether distances, the algorithm often failed to find a satisfying structure. We attribute this to the fact that the solutions for shorter inter-tether distances involve the long double-stranded region and the two single-stranded spacers pointing in opposite directions, so that the spacers can accommodate the fixed-length duplex. This, coupled with the fact that the `least_squares` approach takes only single steps along optimal gradients, makes it prone to getting stuck in local minima for such problems.

Therefore, we tested a second optimization approach, using a global minimizer: the `basinhopping` function from `scipy.optimize`. This algorithm uses larger steps with a Metropolis acceptance criterion to escape local minima, coupled with subsequent local minimization using the BFGS algorithm [46]. We implemented a callback function that immediately exits the solver if a solution is found with energy zero, rather than trying to optimize further. This algorithm does not handle variable constraints directly, so we added these as additional energies: if a variable satisfies the bounds outlined above, its energy is zero, if not, its energy is 10^{100}. This approach was able to find a satisfying structure in 100% of the runs where solutions exist, even with only 10 sampling attempts permitted. We note that here, as in the case of the other algorithms tested, no satisfying structures were found where they were not expected. This shows that our system is better at finding solutions that exist, and does not incorrectly identify any non-solutions as satisfying the constraints.

In summary, these results demonstrate that an optimization-based approach to constraint solving, with a judicious choice of optimization algorithm, can greatly improve the reliability of enumeration results from sampling-based geometric constraint solving.

4.2 Tethered signal transmission line example

We next tested our approach on the tethered signal transmission line system reported by Chatterjee et al. [8]. In this design, a sequence of hairpins are tethered to a DNA origami tile, with a second set of fuel hairpins freely diffusing in solution. That paper reported several transmission lines as well as multi-hairpin logic gates. The two-stator variant of the transmission line system is illustrated in Figure 3, along with the expected reactions. When the first stator is opened via strand displacement by an input strand, it can in turn open a fuel hairpin, which may then bind and open the second stator in the transmission line, and so on. In this system, we consider the two inter-stator distances studied in the original paper: 10.88 nm for the single-spaced variant (also referred to as "1x" spacing below) and 21.76 nm for the double-spaced ("2x") variant. The domain lengths outlined in Figure 3 match those used in the paper, and the system is designed such that an opened stator with a bound found hairpin can interact with the next stator in the single-spaced case but *not* in the double-spaced case. Thus, in the double-spaced case, only the first eight reversible reactions in Figure 3 (those above the broken red line) should be possible. The three-stator system behaves similarly, except that the second stator has an f^* looped toehold rather than a y^* toehold, which means it can only interact with another fuel hairpin. The diffusing, double-stranded reporter complex can only interact with the third stator via *its* y^* toehold.

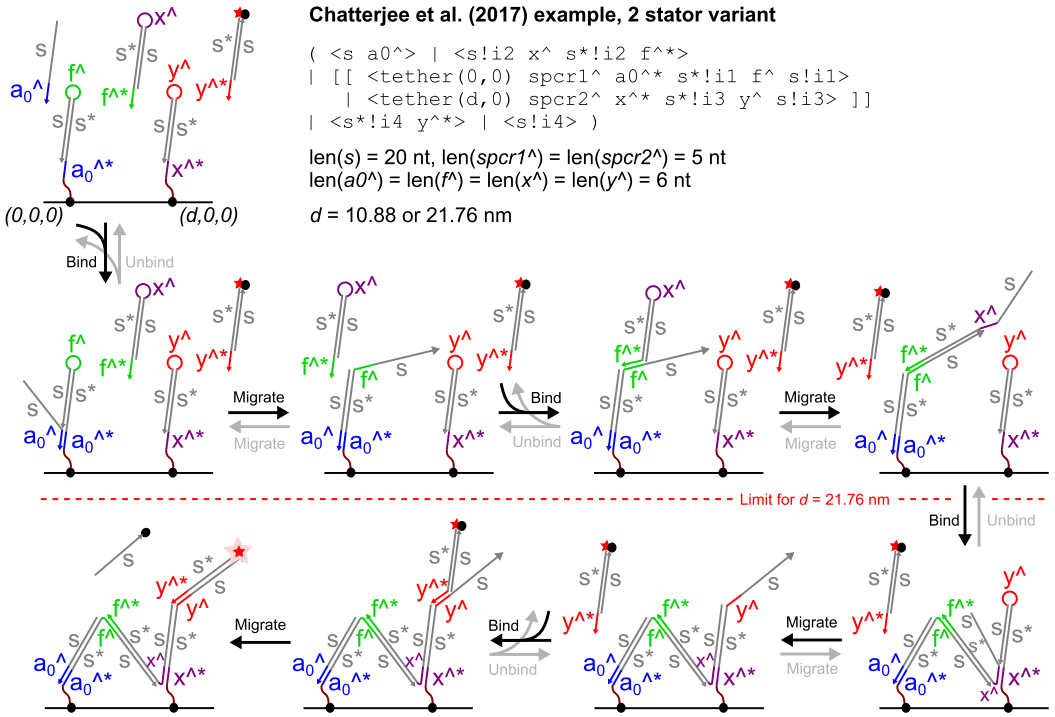

Figure 3 System outline of tethered signal transmission line of Chatterjee et al. [8], including a summary of expected reactions for single- and double-spaced variants of the two-stator system. Domain lengths used here match those used in that paper's experimental work.

We used the same three approaches to constraint solving described in Section 4.1 above. We tested each 100 times on both the two- and three-stator systems, for 10, 25, and 50 attempts. We also compared the single-spaced ("1x") and double-spaced ("2x") variants of each system. In this example there is another parameter to consider: the maximum allowed angle of deviation between two double-stranded regions that meet at a nick. In previous work [25], we introduced this as a parameter that can be used to tune the number of "unexpected" species uncovered by a geometric reaction enumerator such as ours. Here, in all tests we used the value of 105° for the maximum nick angle. In general, larger nick angle parameters are more permissive and lead to more species being enumerated.

We first analyzed the likelihood of each constraint solver to fully enumerate the complete cascade in each case. We expect that the cascade should never complete when the tethered are double-spaced, but for single-spaced tethers both the two- and three-stator systems can theoretically complete. We determined cascade completion by searching the enumerated CRN for an irreversible reaction that displaces the product strand ⟨s⟩: the only way this can happen is if the final stator is opened and then activates the reporter complex. These results are presented in Figure 4, which shows that the sampling-only approach does not successfully enumerate the entire cascade in any case. The optimizer using the `least_squares` algorithm did better on the two-stator system, with over 80% success when it was permitted 50 attempts. However, it failed entirely on the three-stator system, which we attribute to the same problem with local minima that we noted above. This is exacerbated in the three-tether transmission line system because there are more ways for the double-stranded regions to fold back on themselves at nicks and create local minima. By contrast, the `basinhopping` algorithm

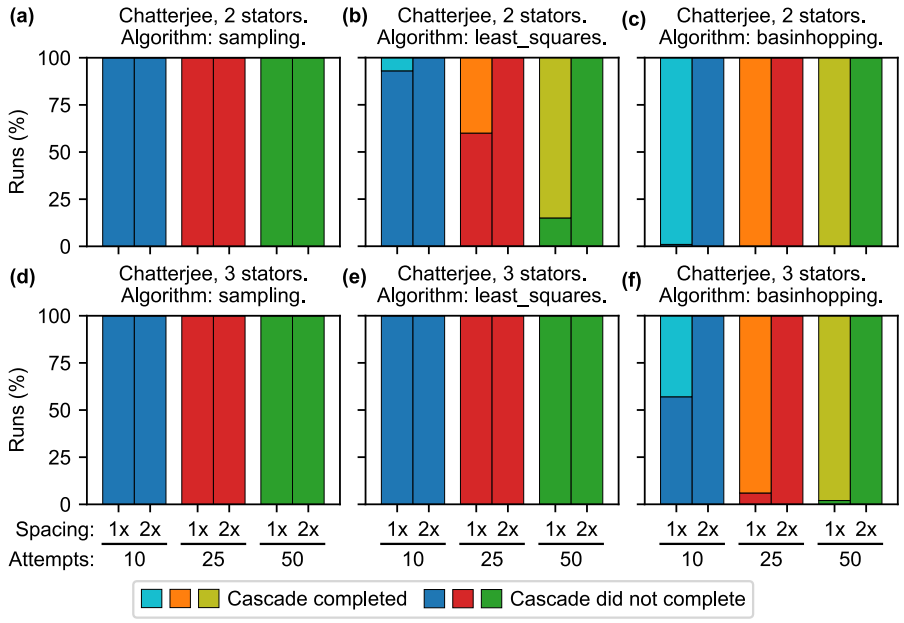

Figure 4 Statistics on cascade completion in 100 attempts to enumerate reaction networks for the Chatterjee et al. tethered transmission line system, outlined in Figure 3. Here and below, results are shown for systems comprising two (**(a)**, **(b)**, **(c)**) or three (**(d)**, **(e)**, **(f)**) stators. As in Figure 2, three approaches are compared: sampling only (**(a)**, **(d)**) and structure optimization using the `least_squares` algorithm (**(b)**, **(e)**) or the `basinhopping` algorithm (**(c)**, **(f)**).

correctly enumerated the entire cascade in virtually all single-spaced cases of the two-stator system. Similar success was seen for the three-stator system when 25 or 50 attempts were allowed. Again, no spurious cascade completions were observed for double-spaced stators.

We next determined how often the expected number of reactions was enumerated; these results are presented in Figure C.3 in the Appendix. In almost all cases, the correct number of reactions was observed in the double-spaced case. For the single-spaced two-stator system, the `least_squares` system outperformed sampling only, and `basinhopping` did better again, with almost 100% success. However, in only around 5% of `basinhopping` tests was the expected number of reactions enumerated for the single-spaced three-stator system.

We sought to explain this result by plotting the distributions of the numbers of reactions enumerated in each case. That data is presented in Figure 5 and shows that the sampling-only and `least_squares` optimizers erred on the side of too few reactions in both the two- and three-stator cases. The `basinhopping` optimizer obtained the expected result in virtually all two-stator tests but found more reactions than expected in most cases for the single-spaced three-stator system. Upon inspection, we found that the third stator enables a number of unexpected reaction products to form, which do not feature in the canonical CRN.

A reaction producing such structures is illustrated in Figure 6(a,b). Even with the maximum nick angle limited to 105°, the three-stator system is able to fold back on itself to carry out such reactions. We note, however, that the fact that many of these species are not always enumerated even by the `basinhopping` algorithm indicates that they may be close to the limit of geometric plausibility, which would in turn suggest that their formation may be low probability events in experiments. Regardless, these results illustrate the potential of our approach for discovering potential unanticipated side-products, so that additional design work to eliminate them may be carried out if needed, prior to experimental testing.

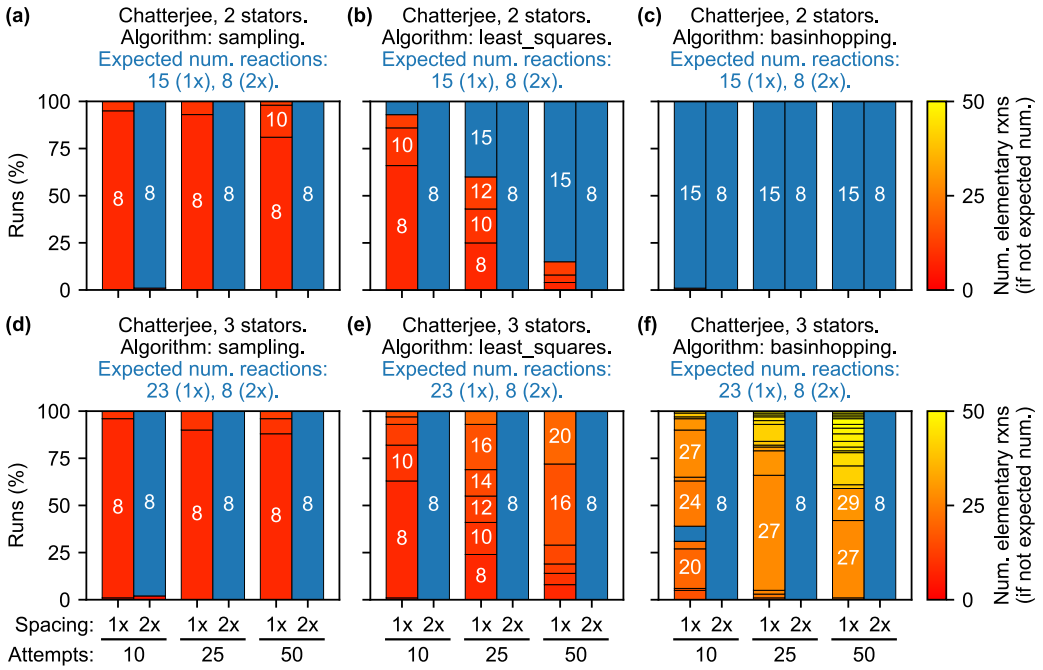

Figure 5 Distributions of numbers of reactions found in 100 attempts to enumerate reaction networks for the Chatterjee et al. tethered transmission line system, outlined in Figure 3. In each case, the numbers of enumeration runs resulting in each number of reactions are plotted as stacked bars, with those producing the smallest number of reactions at the bottom. Bars are color-coded according to the colorbars except for the bar representing the expected number of reactions, which is colored blue. Bars of sufficient size are also labeled with the corresponding number of reactions.

5 Discussion

In summary, we have developed an optimization algorithm that attempts to convert a sampled structure that initially fails to satisfy one or more constraints into one that satisfies them all. Our results show that this can enhance the reliability of geometric reaction enumerators for tethered DNA reaction networks while requiring fewer sampling attempts. This improves upon our previous work in this area [25] and may serve as the basis for future design tools for localized DNA systems, including molecular computers, walkers, and robots.

5.1 Performance Considerations

Here we focused on improving the *reliability* of geometric reaction enumeration. Indeed, our results showed that we can achieve more consistent results with fewer structure sampling attempts than in previous work [22, 25], in particular if the `basinhopping` optimization algorithm is employed. However, the total time taken for enumeration is typically longer, because far more work is carried out in an attempt to salvage each structure sample before it is discarded. We believe that this is a worthwhile tradeoff to obtain a more consistent output from the constraint solver. In this paper we present the statistics from 100 attempts to enumerate each variant of our example systems, though a user might in practice employ some kind of majority voting to settle on a consensus model. Alternatively, a "just-in-time" compilation algorithm such as that employed in our previous work for efficient enumeration and stochastic simulation of very large CRNs [26] or polymerizing CRNs with a potentially infinite number of species and reactions [28] could defer the costs of reaction enumeration.

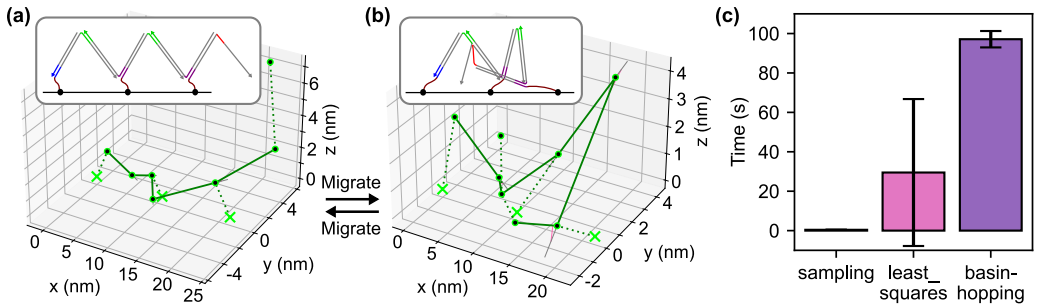

Figure 6 Further analysis of results and performance for the transmission line example of Chatterjee et al. [8]. **(a)**,**(b)** Example of an unintended interaction in the single-spaced, three-stator system. When the third stator hairpin is opened, as in **(a)**, and has a fuel hairpin attached, the `s` domain of the fuel hairpin is able to displace the corresponding domain from the first stator, as in **(b)**. Insets show secondary structure cartoons in a two-dimensional projection; main plots show satisfying 3D geometric structures found by our constraint solver using the `basinhopping` optimization algorithm. In the 3D plots, solid lines are double-stranded regions and broken lines are single-stranded. Crosses are tethers; circles are non-tether vertices. Nick angles are annotated in pink, though the projection means that they are not all visible. **(c)** Average runtimes for all three algorithms applied to the single-spaced, two-stator variant of the tethered transmission line example. In each case, 10 sampling attempts were permitted per species and, as above, the maximum nick angle was 105°. Bars are average runtimes for 100 distinct seeds; errorbars show the standard deviation. Tests were carried out on an Apple MacBook Pro with M1 Pro CPU and 32GB RAM.

Figure 6(c) shows the average time per enumeration for the single-spaced, two-stator transmission line example [8]. The time taken by the sampling-only approach is significantly lower, not just because it does less work but also because it fails to enumerate most of the reactions (Figure 5(a)). The tradeoff for longer runtime is the significant reliability gains from the `basinhopping` approach. While, in the two-stator case, increasing the number of attempts can produce more reliable results from a sampling-only system [25], the same is not true for the three-stator case. Thus, our enhanced approach has significant advantages when we attempt to scale to larger systems involving more tethers, more loops, and more ways to violate constraints. The time taken for the `least_squares` algorithm falls in between as it does less work than the `basinhopping` algorithm; the high variance arises from the fact that for some seeds it fails to enumerate the interaction with the second stator (Figure 5(c)).

Other approaches to improving efficiency could include better structure sampling approaches that look for subgraph isomorphisms between a candidate structure and existing species known to be geometrically plausible, so that the corresponding parts of the new species can begin with their constraints already satisfied. Subdividing structures appropriately could be challenging in general, but splitting based on routes to different tethers might provide a general-purpose approach. We note, however, that even if subgraph constraints can be solved independently, there would still need to be a final integration and verification step to confirm that the structure as a whole is still geometrically plausible. This could yield some gains, though much of the effort would still be spent failing to find solutions for structures that are not geometrically plausible. Therefore, hard-coding short-hand checks for common failure cases, such as looped double-stranded domains, might yield greater improvements.

Because we use the domain-level modeling abstraction common in many design tools [1,32], there is the possibility that an intermediate reaction state, e.g., partially displaced along a duplex domain, might be implausible while the initial and final states are both plausible. Such a reaction would be enumerated by our system even though it could not occur in reality. While theoretically possible, we believe this to be highly unlikely in practice.

5.2 Related and Future Work

The approach developed here draws from a long history of related work on design tools for solution-phase DNA computing systems. A range of such tools exist, including Peppercorn [1], Piperine [42], the Nuskell system [2], KinDA [3], the DyNAMiC Workbench [20], and the Visual DSD system [32, 41] offer a number of options for specifying a system design as a structural description and compiling that description into a kinetic model by enumerating all possible reactions that could occur given those starting species. Considering tethered molecular systems more generally, the work of Dalchau et al. [12] and Dannenberg et al. [13] analyzed specific circuit motifs; the goal of the work outlined here is to generalize and automate such analyses so that they may be applied to arbitrary strand graph reaction systems without specific foreknowledge. We draw inspiration from *distance matrix* approaches [44], though our problem is more complex because single-stranded domains are of variable length.

Our tools could be integrated with DNA sequence designers such as NUPACK [47,48] and Piperine [42], and with DNA origami design tools such as `caDNAno` [16] and `scadnano` [15]. This could enable staple and oligonucleotide sequences for specific designs to be derived, easing the transition to experimental validation of systems that integrate molecular computing and self-assembly, such as signal-passing [35] and reconfigurable DNA structures [18, 49].

Although our results do agree qualitatively with experimental work on the tethered transmission line system [8], further work could involve comparing the predictions from our system with those from other experiments and from coarse-grained models. Integration with other systems, such as oxDNA [4, 17, 34, 37, 43], could inform improvements to the biophysical model that underlies our constraint solving system. We aimed to keep the constraint problems as simple as possible by abstracting away much of the molecular detail: reducing double-strands to rigid rods and neglecting the width of a duplex. However, we could also preserve the rigid rod abstraction while reintroducing some more realistic aspects, e.g., explicitly modeling the width and double-helical structure of duplexes.

We note that our approach does not *guarantee* that all plausible species will be enumerated, but our results support the claim that it greatly increases the probability that if a plausible structure exists, it will be found. Nevertheless, further work will be needed to fully understand the system's sensitivity to settings such as the chosen optimization algorithm and other parameters, such as the nick flexibility limits, and the number of tethers in localized species.

Finally, our definition of a "nick" may need to be refined: if more than two regions intersect, e.g., in a four-way junction, it might prove needlessly restrictive. Furthermore, nicks are in fact extremely flexible: our restrictions on nick angles are thus not necessarily physical but rather a useful modeling tool, enabling the user to focus on core interactions and only dig into more esoteric behaviors when appropriate. Additionally, many of these reactions could also be ruled out via a toehold adjacency criterion [36], though we permit them here so as to not rule out other similar "remote toehold" interactions [19]. This enables a more nuanced analysis of structures that may form via localized strand displacement reactions.

References

1 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. `doi:10.1098/rsif.2019.0866`.

2 Stefan Badelt, Seung Woo Shin, Robert F. Johnson, Qing Dong, Chris Thachuk, and Erik Winfree. A general-purpose CRN-to-DSD compiler with formal verification, optimization, and simulation capabilities. In R. Brijder and L. Qian, editors, *Proceedings of the 23rd International Conference on DNA Computing and Molecular Programming*, volume 10467 of *Lecture Notes in Computer Science*, pages 232–248, 2017. `doi:10.1007/978-3-319-66799-7_15`.

3 Joseph Berleant, Christopher Berlind, Stefan Badelt, Frits Dannenberg, Joseph Schaeffer, and Erik Winfree. Automated sequence-level analysis of kinetics and thermodynamics for domain-level DNA strand-displacement systems. *Journal of the Royal Society Interface*, 15:20180107, 2018. doi:10.1098/rsif.2018.0107.

4 Joakim Bohlin, Michael Matthies, Erik Poppleton, Jonah Procyk, Aatmik Mallya, Hao Yan, and Petr Šulc. Design and simulation of DNA, RNA and hybrid protein–nucleic acid nanostructures with oxView. *Nature Protocols*, 17:1762–1788, 2022. doi:10.1038/s41596-022-00688-5.

5 Hieu Bui, Sudhanshu Garg, Vincent Miao, Tianqi Song, Reem Mokhtar, and John Reif. Design and analysis of linear cascade DNA hybridization chain reactions using DNA hairpins. *New Journal of Physics*, 19:015006, 2017. doi:10.1088/1367-2630/aa53d0.

6 Hieu Bui, Vincent Miao, Sudhanshu Garg, Reem Mokhtar, Tianqi Song, and John Reif. Design and analysis of localized DNA hybridization chain reactions. *Small*, 13(12):1602983, 2017. doi:10.1002/smll.201602983.

7 Hieu Bui, Shalin Shah, Reem Mokhtar, Tianqi Song, Sudhanshu Garg, and John Reif. Localized DNA hybridization chain reactions on DNA origami. *ACS Nano*, 12(2):1146–1155, 2018. doi:10.1021/acsnano.7b06699.

8 Gourab Chatterjee, Neil Dalchau, Richard A. Muscat, Andrew Phillips, and Georg Seelig. A spatially localized architecture for fast and modular DNA computing. *Nature Nanotechnology*, 12:920–927, 2017. doi:10.1038/nnano.2017.127.

9 Yuan-Jyue Chen, Neil Dalchau, Niranjan Srinivas, Andrew Phillips, Luca Cardelli, David Soloveichik, and Georg Seelig. Programmable chemical controllers made from DNA. *Nature Nanotechnology*, 8:755–762, 2013. doi:10.1038/nnano.2013.189.

10 Kevin M. Cherry and Lulu Qian. Scaling up molecular pattern recognition with DNA-based winner-take-all neural networks. *Nature*, 559:370–376, 2018. doi:10.1038/s41586-018-0289-6.

11 Kevin M. Cherry and Lulu Qian. Supervised learning in DNA neural networks. *Nature*, 645:639–647, 2025. doi:10.1038/s41586-025-09479-w.

12 Neil Dalchau, Harish Chandran, Nikhil Gopalkrishnan, Andrew Phillips, and John Reif. Probabilistic analysis of localized DNA hybridization circuits. *ACS Synthetic Biology*, 4(8):898–913, 2015. doi:10.1021/acssynbio.5b00044.

13 Frits Dannenberg, Marta Kwiatkowska, Chris Thachuk, and Andrew J. Turberfield. DNA walker circuits: computational potential, design, and verification. *Natural Computing*, 14:195–211, 2015. doi:10.1007/s11047-014-9426-9.

14 Leonardo de Moura and Nikolaj Bjørner. Z3: An efficient SMT solver. In *Proceedings of TACAS 2008*, volume 4963 of *LNCS*, pages 337–340. Springer, 2008. doi:10.1007/978-3-540-78800-3_24.

15 David Doty, Benjamin L. Lee, and Tristan Stérin. scadnano: A browser-based, scriptable tool for designing DNA nanostructures. In Cody Geary and Matthew J. Patitz, editors, *26th International Conference on DNA Computing and Molecular Programming (DNA 26)*, volume 174 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 9:1–9:17, Dagstuhl, Germany, 2020. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.DNA.2020.9.

16 Shawn M. Douglas, Adam H. Marblestone, Surat Teerapittayanon, Alejandro Vazquez, George M. Church, and William M. Shih. Rapid prototyping of 3D DNA-origami shapes with caDNAno. *Nucleic Acids Research*, 37:5001–5006, 2009. doi:10.1093/nar/gkp436.

17 Jonathan P. K. Doye, Thomas E. Ouldridge, Ard A. Louis, Flavio Romano, Petr Šulc, Christian Matek, Benedict E. K. Snodin, Lorenzo Rovigatti, John S. Schreck, Ryan M. Harrison, and William P. J. Smith. Coarse-graining DNA for simulations of DNA nanotechnology. *Physical Chemistry Chemical Physics*, 15:20395–20414, 2013. doi:10.1039/C3CP53545B.

18 Sisi Fan, Jin Cheng, Yan Liu, Dongfang Wang, Tao Luo, Bin Dai, Chuan Zhang, Daxiang Cui, Yonggang Ke, and Jie Song. Proximity-induced pattern operations in reconfigurable DNA origami domino array. *Journal of the American Chemical Society*, 142(34):14566–14573, 2020. doi:10.1021/jacs.0c06061.

19 Anthony J. Genot, David Yu Zhang, Jonathan Bath, and Andrew J. Turberfield. Remote toehold: A mechanism for flexible control of DNA hybridization kinetics. *Journal of the American Chemical Society*, 133(7):2177–2182, 2011. doi:10.1021/ja1073239.

20 Casey Grun, Justin Werfel, David Yu Zhang, and Peng Yin. DyNAMiC Workbench: an integrated development environment for dynamic DNA nanotechnology. *Journal of the Royal Society Interface*, 12:20150580, 2015. doi:10.1098/rsif.2015.0580.

21 Dejan Jovanović and Leonardo de Moura. Solving non-linear arithmetic. In *Proceedings of IJCAR 2012*, volume 7364 of *LNCS*, pages 339–354. Springer, 2012. doi:10.1007/978-3-642-31365-3_27.

22 Sarika Kumar and Matthew R. Lakin. A geometric framework for reaction enumeration in computational nucleic acid devices. *Journal of the Royal Society Interface*, 20(208):20230259, 2023. doi:10.1098/rsif.2023.0259.

23 Sarika Kumar, Julian M. Weisburd, and Matthew R. Lakin. Structure sampling for computational estimation of localized DNA interaction rates. *Scientific Reports*, 11:12730, 2021. doi:10.1038/s41598-021-92145-8.

24 Matthew R. Lakin. LocalizedEnumeratorOptimization. Software, swhId: swh:1:dir:095f551c8228cf74f1548237675d8391b3963452 (visited on 2026-07-15). URL: https://github.com/matthewlakin/LocalizedEnumeratorOptimization.

25 Matthew R. Lakin and Sarika Kumar. Geometric enumeration of localized DNA strand displacement reaction networks. In Shinnosuke Seki and Jaimie Marie Stewart, editors, *30th International Conference on DNA Computing and Molecular Programming (DNA 30)*, volume 314 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 1:1–1:24, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.DNA.30.1.

26 Matthew R. Lakin, Loïc Paulevé, and Andrew Phillips. Stochastic simulation of multiple process calculi for biology. *Theoretical Computer Science*, 431:181–206, 2012. doi:10.1016/j.tcs.2011.12.057.

27 Matthew R. Lakin, Rasmus Petersen, Kathryn E. Gray, and Andrew Phillips. Abstract modelling of tethered DNA circuits. In Satoshi Murata and Satoshi Kobayashi, editors, *Proceedings of the 20th International Conference on DNA Computing and Molecular Programming*, volume 8727 of *Lecture Notes in Computer Science*, pages 132–147. Springer International Publishing, 2014. doi:10.1007/978-3-319-11295-4_9.

28 Matthew R. Lakin and Andrew Phillips. Modelling, simulating and verifying Turing-powerful strand displacement systems. In L. Cardelli and W. Shih, editors, *Proceedings of the 17th International Conference on DNA Computing and Molecular Programming*, volume 6937 of *Lecture Notes in Computer Science*, pages 130–144. Springer-Verlag, 2011. doi:10.1007/978-3-642-23638-9_12.

29 Matthew R. Lakin and Andrew Phillips. Automated, constraint-based analysis of tethered DNA nanostructures. In R. Brijder and L. Qian, editors, *Proceedings of the 23rd International Conference on DNA Computing and Molecular Programming*, volume 10467 of *Lecture Notes in Computer Science*, pages 1–16. Springer, Cham, 2017. doi:10.1007/978-3-319-66799-7_1.

30 Matthew R. Lakin and Andrew Phillips. Automated analysis of tethered DNA nanostructures using constraint solving. *Natural Computing*, 17(4):709–722, 2018. doi:10.1007/s11047-018-9693-y.

31 Matthew R. Lakin, Simon Youssef, Luca Cardelli, and Andrew Phillips. Abstractions for DNA circuit design. *Journal of the Royal Society Interface*, 9(68):460–486, 2012. doi:10.1098/rsif.2011.0343.

32 Matthew R. Lakin, Simon Youssef, Filippo Polo, Stephen Emmott, and Andrew Phillips. Visual DSD: a design and analysis tool for DNA strand displacement systems. *Bioinformatics*, 27(22):3211–3213, 2011. doi:10.1093/bioinformatics/btr543.

33 Richard A. Muscat, Karin Strauss, Luis Ceze, and Georg Seelig. DNA-based molecular architecture with spatially localized components. In *ISCA '13: Proceedings of the 40th Annual International Symposium on Computer Architecture*, pages 177–188, 2013. doi:10.1145/2485922.2485938.

34 Thomas E. Ouldridge, Rollo L. Hoare, Ard A. Louis, Jonathan P. K. Doye, Jonathan Bath, and Andrew J. Turberfield. Optimizing DNA nanotechnology through coarse-grained modeling: A two-footed DNA walker. *ACS Nano*, 7:2479–2490, 2013. doi:10.1021/nn3058483.

35 Jennifer E. Padilla, Ruojie Sha, Martin Kristiansen, Junghuei Chen, Natasha Jonoska, and Nadrian C. Seeman. A signal-passing DNA-strand-exchange mechanism for active self-assembly of DNA nanostructures. *Angewandte Chemie International Edition*, 54(20):5939–5942, 2015. doi:10.1002/anie.201500252.

36 Rasmus L. Petersen, Matthew R. Lakin, and Andrew Phillips. A strand graph semantics for DNA-based computation. *Theoretical Computer Science*, 632:43–73, 2016. doi:10.1016/j.tcs.2015.07.041.

37 Erik Poppleton, Joakim Bohlin, Michael Matthies, Shuchi Sharma, Fei Zhang, and Petr Šulc. Design, optimization and analysis of large DNA and RNA nanostructures through interactive visualization, editing and molecular simulation. *Nucleic Acids Research*, 48(12):e72, 2020. doi:10.1093/nar/gkaa417.

38 Lulu Qian and Erik Winfree. Scaling up digital circuit computation with DNA strand displacement cascades. *Science*, 332:1196–1201, 2011. doi:10.1126/science.1200520.

39 Lulu Qian, Erik Winfree, and Jehoshua Bruck. Neural network computation with DNA strand displacement cascades. *Nature*, 475:368–372, 2011. doi:10.1038/nature10262.

40 Paul W. K. Rothemund. Folding DNA to create nanoscale shapes and patterns. *Nature*, 440:297–302, 2006. doi:10.1038/nature04586.

41 Carlo Spaccasassi, Matthew R. Lakin, and Andrew Phillips. A logic programming language for computational nucleic acid devices. *ACS Synthetic Biology*, 8(7):1530–1547, 2019. doi:10.1021/acssynbio.8b00229.

42 Niranjan Srinivas, James Parkin, Georg Seelig, Erik Winfree, and David Soloveichik. Enzyme-free nucleic acid dynamical systems. *Science*, 358(6369):eaal2052, 2017. doi:10.1126/science.aal2052.

43 Antonio Suma, Erik Poppleton, Michael Matthies, Petr Šulc, Flavio Romano, Ard A. Louis, Jonathan P. K. Doye, Cristian Micheletti, and Lorenzo Rovigatti. TacoxDNA: A user-friendly web server for simulations of complex DNA structures, from single strands to origami. *Journal of Computational Chemistry*, 40(29):2586–2595, 2019. doi:10.1002/jcc.26029.

44 Michael W. Trosset. Distance matrix completion by numerical optimization. *Computational Optimization and Applications*, 17:11–22, 2000. doi:10.1023/A:1008722907820.

45 Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C. J. Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature Methods*, 17:261–272, 2020. doi:10.1038/s41592-019-0686-2.

46 David J. Wales and Jonathan P. K. Doye. Global optimization by basin-hopping and the lowest energy structures of Lennard-Jones clusters containing up to 110 atoms. *Journal of Physical Chemistry A*, 101(28):5111–5116, 1997. doi:10.1021/jp970984n.

47 Brian R. Wolfe and Niles A. Pierce. Sequence design for a test tube of interacting nucleic acid strands. *ACS Synthetic Biology*, 4(10):1086–1100, 2015. doi:10.1021/sb5002196.

48 Brian R. Wolfe, Nicholas J. Porubsky, Joseph N. Zadeh, Robert M. Dirks, and Niles A. Pierce. Constrained multistate sequence design for nucleic acid reaction pathway engineering. *Journal of the American Chemical Society*, 139(8):3134–3144, 2017. doi:10.1021/jacs.6b12693.

49 Fei Zhang, Jeanette Nangreave, Yan Liu, and Hao Yan. Reconfigurable DNA origami to generate quasifractal patterns. *Nano Letters*, 12:3290–3295, 2012. doi:10.1021/nl301399z.

A Tethered reaction enumeration definitions

$$\frac{G = (V, \mathit{length}, \mathit{color}, A, \mathit{toehold}, E) \qquad a \in A \setminus E \qquad a \cap \mathit{sites}(E) = \varnothing \qquad G' = (V, \mathit{length}, \mathit{color}, A, \mathit{toehold}, (E \cup \{a\})) \qquad \mathit{plausible}(G')}{G \to G'} \text{ (Bind)}$$

$$\frac{G = (V, \mathit{length}, \mathit{color}, A, \mathit{toehold}, E) \qquad a = \{s_1, s_2\} \in E \qquad \mathit{toehold}(a) \qquad \neg\, \mathit{sameSpecies}(s_1, s_2, G') \qquad G' = (V, \mathit{length}, \mathit{color}, A, \mathit{toehold}, (E \setminus \{a\})) \qquad \mathit{plausible}(G')}{G \to G'} \text{ (Unbind)}$$

$$\frac{G = (V, \mathit{length}, \mathit{color}, A, \mathit{toehold}, E) \qquad e = \{s, s'\} \in E \qquad a = \{s, s''\} \in A \setminus E \qquad s'' \notin \mathit{sites}(E) \qquad \mathit{sameSpecies}(s, s'', G) \qquad G' = (V, \mathit{length}, \mathit{color}, A, \mathit{toehold}, ((E \setminus \{e\}) \cup \{a\})) \qquad \mathit{plausible}(G')}{G \to G'} \text{ (Migrate3)}$$

$$\frac{\begin{array}{c} G = (V, \mathit{length}, \mathit{color}, A, \mathit{toehold}, E) \qquad e_1 = \{\mathit{3pr}(s_4), s_1\} \in E \\ e_2 = \{\mathit{3pr}(s_1), s_2\} \in E \qquad e_3 = \{\mathit{3pr}(s_2), s_3\} \in E \qquad e_4 = \{\mathit{3pr}(s_3), s_4\} \in E \\ a_1 = \{\mathit{3pr}(s_1), s_4\} \in A \qquad a_2 = \{\mathit{3pr}(s_3), s_2\} \in A \\ G' = (V, \mathit{length}, \mathit{color}, A, \mathit{toehold}, ((E \setminus \{e_2, e_4\}) \cup \{a_1, a_2\})) \qquad \mathit{plausible}(G') \end{array}}{G \to G'} \text{ (Migrate4)}$$

Figure A.1 Inference rules defining a geometrically-aware transition relation ($\to$) between strand graphs; reproduced from previous work [22] for convenience. The *plausible* predicate defines the interface to a geometric constraint solver that which determines if the reaction is permitted by the geometry of the candidate product structure.

B Tethered region graph definitions

Here we reproduce the definition of region graphs from our previous work [22]. We do not modify this to incorporate tether information as it is not necessary to do so; tether information is added back when the constraints to be solved for a given tethered strand graph are defined.

B.1 Defining regions

In previous work, we defined a notion of "condensing" a structure representation to collapse adjacent single-stranded or double-stranded domains in a process representation of a structure, so that the domain boundaries became the only important positions represented in the constraint problem. Here we use a similar idea by assigning domains to "regions." Each region is assigned a label, which we represent using natural numbers.

$$\frac{S_i \in \overline{S} \qquad S_i \to G' \qquad \overline{S}' = \overline{S} \setminus \{S_i\} \cup \mathit{tethered}(G')}{[[\,\overline{S}\,]] \Rightarrow [[\,\overline{S}'\,]] + \mathit{free}(G')} \text{ (TileOne)}$$

$$\frac{S_i, S_j \in \overline{S} \qquad i \neq j \qquad (S_i \mid\mid S_j) \to G' \qquad \overline{S}' = \overline{S} \setminus \{S_i, S_j\} \cup \mathit{tethered}(G')}{[[\,\overline{S}\,]] \Rightarrow [[\,\overline{S}'\,]] + \mathit{free}(G')} \text{ (TileTwo)}$$

$$\frac{S_i \in \overline{S} \qquad (S_i \mid\mid S) \to G' \qquad \overline{S}' = \overline{S} \setminus \{S_i\} \cup \mathit{tethered}(G')}{[[\,\overline{S}\,]] + S \Rightarrow [[\,\overline{S}'\,]] + \mathit{free}(G')} \text{ (TileFree)}$$

$$\frac{(S \mid\mid S') \to G' \qquad \mathit{tethered}(G') = \varnothing}{S + S' \Rightarrow \mathit{free}(G')} \text{ (FreeFree)}$$

Figure A.2 Inference rules defining a geometrically-aware transition relation ($\Rightarrow$) between collections of strand graph species, including tile species with tethered components; these are reproduced from previous work [25] for convenience. These rules build on our previously reported $\to$ rules for geometrically aware reaction enumeration on strand graphs of free species [22], shown here in Figure A.1. The rules presented here implement a CRN semantics [36] for systems that may contain both free and tethered species, where we write $\overline{S}$ for finite collections of fully connected strand graphs, each of which represents a single molecular species. The syntax $[[\,\overline{S}\,]]$ represents a tile species containing species $\overline{S}$, each of which must be tethered; we assume that other species are freely diffusing. We write $S \mid\mid S'$ for the strand graph resulting from the parallel composition of S and S'. We write $\mathit{tethered}(G)$ and $\mathit{free}(G)$ for the connected components of a strand graph G that represent tethered and free species, that is, those species which do or do not include a tether specification, respectively. Rule (TileOne) identifies a single species tethered to the same tile that may react. Similarly, rule (TileTwo) encodes a reaction between two species tethered to the same tile. Rule (TileFree) identifies a free species that may react with a species tethered to a tile, and rule (FreeFree) lifts the standard rules for reactions involving two free species into the $\Rightarrow$ relation. All reactions involving at least one tile species may produce new tethered species, and possibly some free species also. In the case of rule (FreeFree), we note that the $\mathit{tethered}(G') = \varnothing$ premise is vacuously true since a tethered species cannot spontaneously emerge from a reaction between two free species. These rules form the basis of our localized geometric reaction enumerator.

▶ **Definition B.1** (Region mappings). *Assuming the definition of strand graphs in the main text, a region mapping for a given strand graph is a function* $\mathit{region} : \mathit{legal}_S \to \mathbb{N}$ *that assigns a region label to every legal site in the strand graph* G*, and which has the following properties:*

- *if* $\mathit{region}((v,n)) = r$ *and* $(v,n) \notin \mathit{sites}(E)$*, then:*
  - *if* $(v, n-1) \in \mathit{legal}_S(V, \mathit{length})$ *then* $\mathit{region}((v,n-1)) = r$ *iff* $(v, n-1) \notin \mathit{sites}(E)$*;*
  - *if* $(v, n+1) \in \mathit{legal}_S(V, \mathit{length})$ *then* $\mathit{region}((v,n+1)) = r$ *iff* $(v, n+1) \notin \mathit{sites}(E)$*.*
- *if* $\mathit{region}((v,n)) = r$ *and* $\{(v,n),(v',n')\} \in E$*, then:*
  - *if* $(v, n-1) \in \mathit{legal}_S(V, \mathit{length})$ *then* $\mathit{region}((v,n-1)) = r$ *iff* $\{(v,n-1),(v',n'+1)\} \in E$*;*
  - *if* $(v, n+1) \in \mathit{legal}_S(V, \mathit{length})$ *then* $\mathit{region}((v,n+1)) = r$ *iff* $\{(v,n+1),(v',n'-1)\} \in E$*.*
- *if we define* $\mathit{allregionlabels}(G) = \{r \mid \exists s \in \mathit{legal}_S.\, \mathit{region}(s) = r\}$*, that is, the set of region labels assigned for strand graph* G*, and there are* n *such distinct region labels, then* $\mathit{allregionlabels}(G) = \{0, \ldots, n-1\}$*.*
- *if* $s < s'$ *in the ordering of sites on the canonically relabeled strand graph and* $\mathit{region}(s) \neq \mathit{region}(s')$ *then* $\mathit{region}(s) < \mathit{region}(s')$*.*

In other words, a valid region mapping assigns the same region label to all sites representing domains in a continuous run of single-stranded domains or double-stranded domains that form a continuous double helix (i.e., with no nicks or overhangs). Furthermore, we note that the sites in a given region are either all bound or all unbound. This allows us to define a predicate *isBoundRegion* on region labels such that $\mathit{isBoundRegion}(r)$ returns true if all sites in region r are bound and false otherwise (i.e., if all sites are unbound). This will be helpful in defining the constraints associated with a region, below. The last two conditions do ensure that the region mapping for a given strand graph is unique; the final condition essentially says that the ordering on sites induces an ordering on regions.

We also define $\mathit{regionsites}(r)$, the inverse of the *region* function, which returns the set of sites contained in a particular region, as follows:

$$\mathit{regionsites}(r) = \{s \mid \mathit{region}(s) = r\}.$$

We define a *position* p as a pair (s, w) where $s \in \mathit{legal}_S$ is a legal site from the original strand graph G and $w \in \{5', 3'\}$ is a tag which determines whether we are referring to the 5' or 3' end of the domain represented by that site. This will be useful below, to define which parts of the structure we are referring to. We will write $\mathit{5prPosns}(r)$ for the set of positions located at the end of region r on the 5' end of their site, and likewise $\mathit{3prPosns}(r)$ for the set of positions located at the end of that region but on the 3' end of their site, which can be defined as follows:

$$\mathit{5prPosns}(r) = \{((v,n), 5') \mid (v,n) \in \mathit{regionsites}(r) \wedge (v,n-1) \notin \mathit{regionsites}(r)\}$$
$$\mathit{3prPosns}(r) = \{((v,n), 3') \mid (v,n) \in \mathit{regionsites}(r) \wedge (v,n+1) \notin \mathit{regionsites}(r)\}.$$

Furthermore, it will also be necessary to uniquely identify the two "ends" of a region r, and compute which positions are located there (there will be one position at each end for a single-stranded region but two at each end for a double-stranded region). For clarity we will refer to these as the A and B ends of the region. These will be defined by exploiting the underlying ordering on sites that arises from the canonical representations defined for strand graphs; in the case of a double-stranded region, this allows us to pick one strand whose directionality then uniquely induces the choice of A and B ends. We will write $\mathit{APosns}(r)$ and $\mathit{BPosns}(r)$ for the sets of positions located at the A and B ends of region r, which can be defined as follows:

- $\mathit{APosns}(r) = \{((v,n), 5')\} \cup \{((v',n'), 3') \mid \{(v,n),(v',n')\} \in E\}$, where (v,n) is the **least** site in the set $\mathit{5prPosns}(r)$.
- $\mathit{BPosns}(r) = \{((v,n), 3')\} \cup \{((v',n'), 5') \mid \{(v,n),(v',n')\} \in E\}$, where (v,n) is the **least** site in the set $\mathit{3prPosns}(r)$.

Finally, we define the set of all positions that occur at either end of some region in G:

$$\mathit{allPosns}(G) = \{p \mid \exists r \in \mathit{allregionlabels}(G).\, p \in (\mathit{5prPosns}(r) \cup \mathit{3prPosns}(r))\}.$$

B.2 Defining region graphs

▶ **Definition B.2** (Region graphs). *Given a (canonically labeled) strand graph G, we are now in a position to define the corresponding region graph, $\mathit{regiongraph}(G)$. The region graph will be an undirected graph (RGV, RGE) whose set of vertices $RGV = \{P_1, \ldots, P_k\}$, where $\{P_1, \ldots, P_k\}$ is a partition of $\mathit{allPosns}(G)$ such that p and p' occur in the same subset P_i iff any of the following statements is true of p and p':*

- *$p \in (\mathit{APosns}(r) \cap \mathit{5prPosns}(r))$ and $p' \in (\mathit{APosns}(r) \cap \mathit{3prPosns}(r))$, for some r.*
- *$p \in (\mathit{BPosns}(r) \cap \mathit{5prPosns}(r))$ and $p' \in (\mathit{BPosns}(r) \cap \mathit{3prPosns}(r))$, for some r.*

- $p = ((v, n), 5')$ *and* $p' = ((v, n-1), 3')$.
- $p = ((v, n), 3')$ *and* $p' = ((v, n+1), 5')$.

Note that the first two points imply that for all $r \in allregionlabels(G)$, $APosns(r) \subseteq P_i$ for some P_i and $BPosns(r) \subseteq P_j$ for some P_j. Note that it is possible for i and j to be equal in the previous statement: this would correspond to a loop comprising a single region. Note also that the latter two points only apply where both p and p' are elements of $allPosns(G)$. We can therefore define the set of edges RGE of the region graph, as follows:

$$RGE = \{\{P_i, P_j\} \mid \exists r \in allregionlabels(G).\, APosns(r) \subseteq P_i \wedge BPosns(r) \subseteq P_j\}.$$

We also assume that each edge is labeled with the corresponding region label r from above; this allows properties of the region such as the length or type (single-stranded vs double-stranded) to be looked up or stored.

Note that, since the region mapping is unique for a given strand graph, the partition used to label the vertices of the region graph will also be unique. Thus, it follows that the region graph $regiongraph(G)$ for a given strand graph G will be unique.

C Additional data

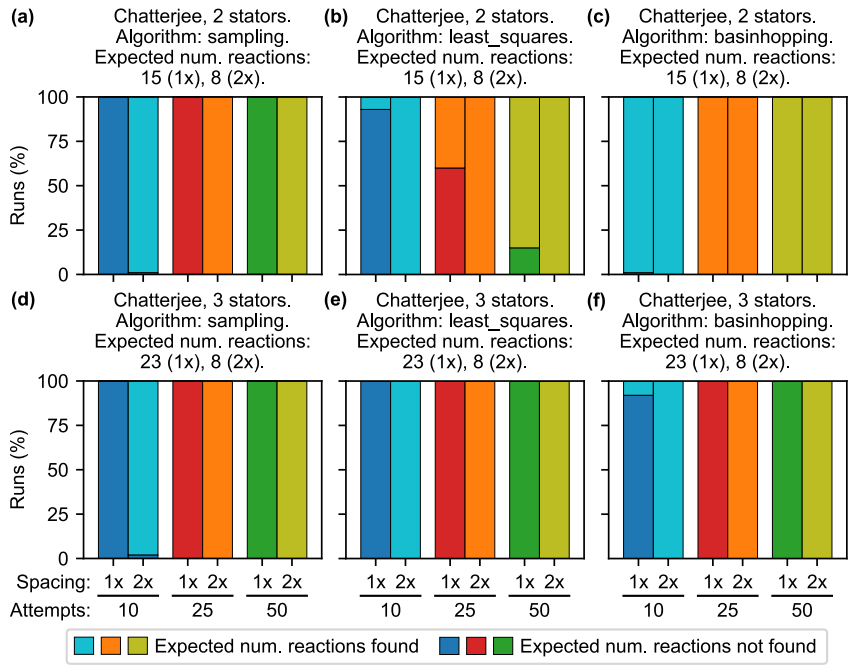

Figure C.3 Statistics on how often we observed the expected number of reactions in 100 attempts to enumerate reaction networks for the Chatterjee et al. tethered transmission line system. The expected numbers of reactions are shown in each subfigure title and were determined via manual analysis of the systems. The analysis for the two-stator case is shown in Figure 3. The single-spaced three-stator case is similar, except that more reactions are required to open the second stator and second fuel hairpin. The expected number of reactions for the double-spaced case is the same for the two- and three-stator systems, as in both cases only the reversible reactions that precede the interaction with the second stator are permitted by the tether geometry.