Stochastic Simulation of the Kinetics of Multiple Interacting Nucleic Acid Strands

Joseph Malcolm Schaeffer[1,2], Chris Thachuk[1], and Erik Winfree[1(✉)]

[1] California Institute of Technology, Pasadena, USA
winfree@caltech.edu
[2] Autodesk Research, San Francisco, USA

Abstract. DNA nanotechnology is an emerging field which utilizes the unique structural properties of nucleic acids in order to build nanoscale devices, such as logic gates, motors, walkers, and algorithmic structures. Predicting the structure and interactions of a DNA device requires effective modeling of both the thermodynamics and the kinetics of the DNA strands within the system. The kinetics of a set of DNA strands can be modeled as a continuous time Markov process through the state space of all secondary structures. The primary means of exploring the kinetics of a DNA system is by simulating trajectories through the state space and aggregating data over many such trajectories. We expand on previous work by extending the thermodynamics and kinetics models to handle multiple strands in a fixed volume, in a way that is consistent with previous models. We developed data structures and algorithms that allow us to take advantage of local properties of secondary structure, improving the efficiency of the simulator so that we can handle reasonably large systems. Finally, we illustrate the simulator's analysis methods on a simple case study.

1 Introduction

Dynamic DNA nanotechnology [29] is an emerging field that utilizes the unique structural properties of nucleic acids in order to build nanoscale devices, such as conformational motors [27], hybridization catalysts [21], logic gates [14,19], analog circuits [2,26,30], triggered self-assembly [6,26], polymerization motors [22], molecular walkers [13,20], and molecular robots [9,12] that operate even in the absence of enzymes and other sophisticated non-nucleic-acid chemistry. These devices are built out of DNA strands whose sequences have been carefully designed in order to control their secondary structure—the hydrogen bonding state of the bases within the strand (called "base-pairing"). This base-pairing is used to not only control the physical structure of the device, but also to enable specific interactions between different components of the system, such as allowing, for example, a DNA strand that catalytically triggers the assembly of two components. Predicting the structure and interactions of a DNA device requires effective modeling of both the thermodynamics and the kinetics of the DNA strands within the system. Thermodynamic models can be used to make

A. Phillips and P. Yin (Eds.): DNA 2015, LNCS 9211, pp. 194–211, 2015.
DOI: 10.1007/978-3-319-21999-8_13

equilibrium predictions for these systems, allowing us to look at questions like "Is the assembled end-product a well-formed molecular structure, and is it energetically favorable?", while kinetics models allow us to predict the non-equilibrium dynamics, such as "How quickly will the catalytic pathway take place?" Although the thermodynamics of multiple interacting DNA strands is a well-studied model [3,4], which allows for both analysis and design of DNA devices [5,28], previous work on secondary structure kinetics models only explored the kinetics of how a single strand folds on itself [7,25].

The kinetics of a set of DNA strands can be modeled as a continuous time Markov process through the state space of all secondary structures. Due to the exponential size of this state space it is computationally intractable to obtain an analytic solution for most problem sizes of interest. Thus the primary means of exploring the kinetics of a DNA system is by simulating trajectories through the state space and aggregating data over many such trajectories. We present here the **Multistrand** kinetics simulator, which extends previous work [7] by using the multiple strand thermodynamics model [4] (a core component for calculating transition rates in the kinetics model), adding new terms to the thermodynamics model to account for stochastic modeling considerations, and by adding new kinetic moves that allow bimolecular interactions between strands. In Ref. [18], we prove that this new kinetics and thermodynamics model is consistent with the prior work on multiple strand thermodynamics models [4].

The Multistrand simulator is based on the Gillespie algorithm [8] for generating statistically correct trajectories of a stochastic Markov process. We developed data structures and algorithms that take advantage of local properties of secondary structures. These algorithms enable the efficient reuse of the basic objects that form the system, such that only a very small part of the state's neighborhood information needs to be recalculated with every step. A key addition was the implementation of algorithms to handle the new kinetic steps that occur between different DNA strands, without increasing the time complexity of the overall simulation. These improvements lead to a reduction in worst case time complexity of a single step and also lead to additional improvements in the average case time complexity.

What data does the simulation produce? At the very simplest, the simulation produces a full kinetic trajectory through the state space—the exact states it passed through, and the time at which it reached them. A small system might produce trajectories that pass through hundreds of thousands of states, and that number increases rapidly as the system gets larger. Going back to our original question, the type of information a researcher hopes to get out of the data could be very simple: "How quickly will the catalytic pathway take place?", with the implied question of whether it's worth it to actually purchase the particular DNA strands composing the catalyst system and perform an experiment, or go back to the drawing board and redesign the device. One way to acquire that type of information is to look at the first time in the trajectory where we reached the "assembly has been catalyzed" state, and record that information for a large number of simulated trajectories in order to obtain a useful answer. We designed and implemented new simulation modes that allow the full trajectory data to be

condensed during generation into only the pieces the user cares about for their particular question. This analysis tool also required the development of flexible ways to talk about states that occur in trajectory data; if someone wants data on when or how the catalyst acted, we have to be able to express that in terms of the Markov process states which meet that condition.

2 The Model

2.1 System Specification

We are interested in simulating nucleic acid molecules (DNA or RNA) in a stochastic regime; that is to say that we have a discrete number of molecules in a fixed volume. This regime is found in experimental systems that have a small volume with a fixed count of each molecule present, such as the interior of a cell, protocell, or droplet. We can also apply this to experimental systems with a larger volume (such as a test tube) when the system is well mixed, as we can either simulate a fixed small volume with small molecular counts and extrapolate to the larger volume, or we can individually simulate the interactions between specific molecules and derive rate constants for a coarse-grained chemical reaction network model that can be simulated in the mass-action regime.

To discuss the modeling and simulation of the system, we begin by defining the components of the system, and what comprises a state of the system within the simulation.

Strands. Each DNA molecule to be simulated is represented by a *strand*. Our system then contains a set of strands Ψ^*, where each strand $s \in \Psi^*$ is defined by $s = (id, label, sequence)$. A strand's *id* uniquely identifies the strand within the system, while the *sequence* is the ordered list of nucleotides that compose the strand. The strand label will usually be ignored, but may be used to make a distinction between strands with identical sequences. For example, if one strand were to be labeled with a fluorophore, it would no longer be physically identical to another with the same sequence but no fluorophore. We define two strands as being *identical* if they have the same labels and sequences.

Complex Microstate. A *complex* is a set of strands connected by base pairing (secondary structure). We define the state of a complex by $c = (ST, \pi^*, BP)$, called the *complex microstate*. The components are a nonempty set of strands $ST \subseteq \Psi^*$, an ordering π^* on the strands ST, and a list of base pairings $BP = \{(i_j \cdot k_l) \mid$ base i on strand j is paired to base k on strand l, and $j \leq l$, with $i < k$ if $j = l\}$, where "strand l" refers to the strand occurring in position l in the ordering π^*. Further, not all base pairings are allowed: following Ref. [4], every complex must by definition be connected, hairpins must have loop lengths of at least three, and in this work only non-pseudoknotted secondary structures will be considered.

System Microstate. A system microstate represents the configuration of the strands in the volume we are simulating (the "box"). We define a *system*

microstate i as a set of complex microstates, such that each strand in the system is in exactly one complex within the system.

2.2 Energy

The conformation of a nucleic acid strand at equilibrium can be predicted by a well-studied model, called the nearest neighbor energy model [15–17]. Recent work has extended this model to cover systems with multiple interacting nucleic acid strands [4]. The distribution of system microstates at equilibrium is a Boltzmann distribution, where the probability of observing a microstate i is given by

$$Pr(i) = \frac{1}{Q_{kin}} * e^{-\Delta G_{box}(i)/RT} \tag{1}$$

where $\Delta G_{box}(i)$ is the free energy of the system microstate i, and is the key quantity determined by these energy models. $Q_{kin} = \sum_i e^{-\Delta G_{box}(i)/RT}$ is the partition function of the system, R is the gas constant, and T is the temperature of the system in Kelvin.

Energy of a System Microstate. To treat the energy of the system microstate i, we break it down into components. The system consists of one or more complex microstates c, each with their own energy. Additionally, the system has an entropy that accounts for the possible spatial arrangements of complexes within the "box".

Let us first consider the entropy term. Our reference state, which by definition will have zero energy, is chosen to be the system microstate in which all strands are in separate complexes and have no base pairs formed. Therefore, our entropy term is in terms of the reduction of available positional states caused by having strands join together. Assuming that the solution is sufficiently dilute that boundary and crowding effects can be ignored (i.e. each complex's center of mass can be anywhere within the simulated volume V), then each complex contributes $RT \log \frac{V}{V_0}$ to the energy of the system, where V_0 is the reference volume[1] chosen to be consistent with existing thermodynamic models. If the system contains L_{tot} strands within a total of C complexes, and we define $\Delta G_{volume} = RT \log \frac{V}{V_0}$, then the contribution to the energy of the system microstate i from the translational entropy of the box, relative to the reference state, is simply $(L_{tot} - C) * \Delta G_{volume}$.

And thus in terms of $C, L_{tot}, \Delta G_{volume}$ and $\overline{\Delta G}(c)$ (the energy of complex microstate c, defined in the next section), we define $\Delta G_{box}(i)$, the energy of the system microstate i, as follows:

[1] We calculate V_0 as the volume in which we would have exactly one molecule at a standard concentration of 1 mol/L: $V_0 = 1/(N_a * 1\,\text{mol/L})$, where N_a is Avogadro's number, and thus V_0 is in liters. Similarly, we may wish to calculate V based on the concentration u in mol/L of a single strand such that the volume V is chosen such that exactly one molecule is present in that volume. In this case we have $V = \frac{1}{u * N_a}$ and the relative number of states in the box is then $\frac{V}{V_0} = \frac{N_a}{u * N_a} = \frac{1}{u}$.

$$\Delta G_{box}(i) = (L_{tot} - C) * \Delta G_{volume} + \sum_{c \in i} \overline{\Delta G}(c) \tag{2}$$

The energy formulas derived here, suitable for our stochastic model, differ from those in [4] in two main ways: the lack of "symmetry terms", and the addition of the ΔG_{volume} term.

Energy of a Complex Microstate. We previously defined a complex microstate in terms of the list of base pairings present within it. However, the well-studied models are based upon nearest neighbor interactions between the nucleic acid bases. These interactions divide the secondary structure of the system into local components which we refer to as *loops*, shown in Fig. 1.

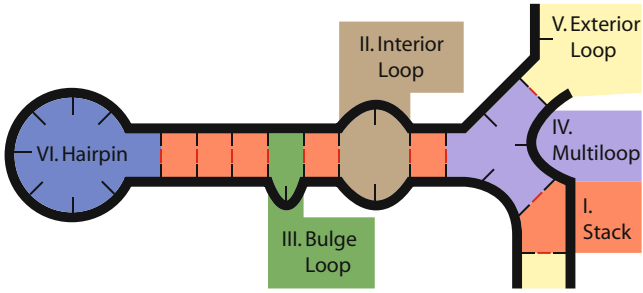


Fig. 1. Secondary structure divided into loops.

These loops can be broken down into different categories, and parameter tables and formulas for each category have been determined from experimental data [17]. Each loop l has an energy, $\Delta G(l)$, which can be retrieved from the appropriate parameter table for its category. Each complex also has an energy contribution associated with the entropic initiation cost [1] (e.g., rotational) of bringing two strands together, ΔG_{assoc}, whose total contribution is proportional to the number of strands L within the complex, as follows: $(L-1) * \Delta G_{assoc}$.

The energy of a complex microstate c is then the sum of these two types of contributions. We can also divide any free energy ΔG into the enthalpic and entropic components, ΔH and ΔS related by $\Delta G = \Delta H + T * \Delta S$. For a complex microstate, each loop can have both enthalpic and entropic components, but ΔG_{assoc} is usually assumed to be purely entropic [16]. This becomes important when determining the kinetic rates, in Sect. 2.3.

We use $\overline{\Delta G}(c)$ to refer to the energy of a complex microstate to be consistent with the nomenclature in [4], where $\overline{\Delta G}(c)$ refers to the energy of a complex when all strands within it are considered unique (as is the case in our system), and $\Delta G(c)$ is the energy of the complex, without assuming that all strands are unique (and thus it must account for rotational symmetries). In summary,

the standard free energy of a complex microstate c, containing $L(c) = |ST(c)|$ strands, is

$$\overline{\Delta G}(c) = \left(\sum_{\text{loop } l \in c} \Delta G(l) \right) + (L(c) - 1)\Delta G_{assoc}$$

which can now be used in combination with Eq. 2 to compute $\Delta G_{box}(i)$.

It can also be convenient to write the system energy as a single sum over the complexes, rather than separating the complex microstate energies and the overall translational entropy terms. Using $L_{tot} = \sum_{c \in i} L(c)$, and $C = \sum_{c \in i} 1$, we obtain

$$\Delta G_{box}(i) = \sum_{c \in i} \left(\overline{\Delta G}(c) + (L(c) - 1) * \Delta G_{volume} \right) \stackrel{def}{=} \sum_{c \in i} \Delta G^*(c)$$

where

$$\begin{aligned} \Delta G^*(c) &= \overline{\Delta G}(c) + (L(c) - 1) * \Delta G_{volume} \\ &= \left(\sum_{\text{loop } l \in c} \Delta G(l) \right) + (L(c) - 1) * (\Delta G_{assoc} + \Delta G_{volume}) \end{aligned}$$

is the component of the total system energy that is associated with complex microstate c.

In Ref. [18], we have compared the Multistrand stochastic energy model to the NUPACK mass action model, showing that despite the lack of symmetry terms and the addition of ΔG_{volume} terms they nonetheless make identical equilibrium predictions.

2.3 Kinetics

Basics. Thermodynamic predictions have only limited use for some systems of interest, if the key information to be gathered is the reaction rates and not the equilibrium states. Many systems have well-defined ending states that can be found by thermodynamic prediction, but predicting whether it will reach the end state in a reasonable amount of time requires modeling the kinetics. Kinetic analysis can also help uncover poor sequence designs, such as those with alternate reactions leading to the same states, or kinetic traps which prevent an intended reaction from occurring quickly.

The kinetics are modeled as a continuous time Markov process over secondary structure space. System microstates i, j are considered adjacent if they differ by a single base pair (Fig. 2), and we choose the transition rates k_{ij} (the transition from state i to state j) and k_{ji} such that they obey detailed balance:

$$\frac{k_{ij}}{k_{ji}} = e^{-\frac{\Delta G_{box}(j) - \Delta G_{box}(i)}{RT}} \tag{3}$$

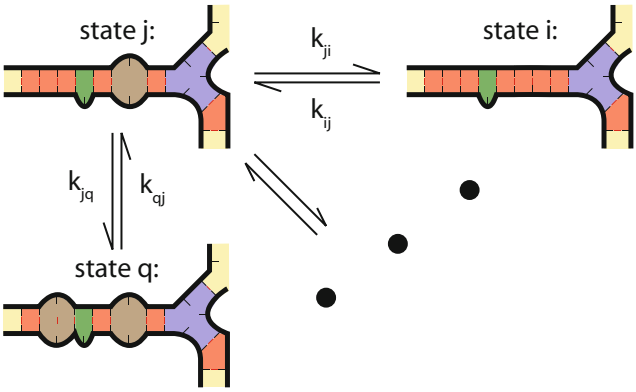


Fig. 2. System microstates i, q adjacent to current state j, with many others not shown.

This property ensures that given sufficient time we will arrive at the same equilibrium state distribution as the thermodynamic prediction (i.e., the Boltzmann distribution on system microstates, Eq. 1) but it does not fully define the kinetics as only the ratio $\frac{k_{ij}}{k_{ji}}$ is constrained. We discuss how to choose these transition rates in the following sections, but regardless of this choice we can still determine how the next state is chosen and the time at which that transition occurs.

Given that we are currently in state i, the next state m in a simulated trajectory is chosen randomly among the adjacent states j, weighted by the rate of transition to each.

$$Pr(m) = \frac{k_{im}}{\Sigma_j k_{ij}} \tag{4}$$

Similarly, the time taken to transition to the next state is chosen randomly from an exponential distribution with rate parameter λ, where λ is the total rate out of the current state, $\Sigma_j k_{ij}$.

$$Pr(\Delta t) = \lambda \exp(-\lambda \Delta t) \tag{5}$$

We will now classify transitions into two exclusive types: those that change the number of complexes present in the system, called *bimolecular transitions*, and those where changes are within a single complex, called *unimolecular transitions*. Note that this terminology is slightly different from the standard use of *bimolecular reactions* and *unimolecular reactions* in chemical reaction network theory: a bimolecular transition could be either a bimolecular reaction (two complexes coming together) or the corresponding unimolecular reaction (one complex dissociating into two).

Unimolecular Transitions. Because unimolecular transitions involve only a single complex, it is natural to define these transitions in terms of the complex microstate which changed, rather than the full system microstate. Like Fig. 2 implies, we define a complex microstate d as being adjacent to a complex microstate c if it differs by exactly one base pair. We call a transition from c

to d that adds a base pair a *creation move*, and a transition from c to d that removes a base pair a *deletion move*. The exclusion of pseudoknotted structures is not inherent in this definition of adjacent states, but rather arises from our disallowing pseudoknotted complex microstates.

The formal Markov chain for Multistrand simulations consists of transitions between system microstates i and j that differ by exactly one base pair, thus any unimolecular transition involves exactly one complex. Note that if i to j is a creation move, j to i must be a deletion move, and vice versa. Similarly, if there is no transition from i to j, there cannot be a transition from j to i, which implies that every unimolecular move in this system is reversible.

Bimolecular Transitions. A bimolecular transition from system microstate i to system microstate j is one where the single base pair difference between them leads to a differing number of complexes within each system microstate. This differing number of complexes could be due to a base pair joining two complexes in i to form a single complex in j, which we will call a *join move*. Conversely, the removal of this base pair from i could cause one complex in i to break into two complexes within j, which we will call a *break move*. Note that if i to j is a join move, then j to i must be a break move, and vice versa. As we saw before, this also implies that every bimolecular move is reversible. Again, while arbitrary bimolecular transitions are not inherently prevented from forming pseudoknots in this model, we implicitly prevent them by using only complex microstates that are not pseudoknotted.

Transition Rates. A key part of our model is the choice of *rate method*: the way we set the rates of a pair of reactions so that they obey detailed balance. There are several rate methods found in the literature [10,11,31] which have been used for kinetics models for single-stranded nucleic acids [7,31] with various energy models. We have implemented two of these simple rate methods which were previously used in single base pair elementary step kinetics models for single stranded systems.

In order to maintain consistency with known thermodynamic models, each pair of k_{ij} and k_{ji} must satisfy detailed balance and thus their ratio is determined by the thermodynamic model, but in principle each pair could be independently scaled by some arbitrary prefactor, perhaps chosen to optimize agreement with experimental results on nucleic acid kinetics. However, since the number of microstates is exponential, this leads to far more model parameters (the prefactors) than is warranted by available experimental data. For the time being, we limit ourselves to using only two scaling factors: k_{uni} for use with unimolecular transitions, and k_{bi} for bimolecular transitions.

Unimolecular Rate Models. The first rate model we will examine is the Kawasaki method [10]. This model has the property that both "downhill" (energetically favorable) and uphill transitions scale directly with the steepness of their slopes.

$$k_{ij} = k_{uni} * e^{-\frac{\Delta G_{box}(j) - \Delta G_{box}(i)}{2RT}} \tag{6}$$

$$k_{ji} = k_{uni} * e^{-\frac{\Delta G_{box}(i) - \Delta G_{box}(j)}{2RT}} \tag{7}$$

The second rate model under consideration is the Metropolis method [11]. In this model, all downhill moves occur at the same fixed rate, and only the uphill moves scale with the slope. This means that the maximum rate for any move is bounded, and in fact all downhill moves occur at this rate. This is in direct contrast to the Kawasaki method, where there is no bound on the maximum rate. For microstates i and j such that $\Delta G_{box}(i) \geq \Delta G_{box}(j)$:

$$k_{ij} = 1 * k_{uni} \tag{8}$$

$$k_{ji} = k_{uni} * e^{-\frac{\Delta G_{box}(i) - \Delta G_{box}(j)}{RT}} \tag{9}$$

Note that the value of k_{uni} that best fits experimental data is likely to be different for both models. Additionally, note that full calculation of $\Delta G_{box}(i)$ and $\Delta G_{box}(j)$ is not necessary in order to calculate the rates, because microstates i and j differ in exactly one pair of complex microstates ($c \in i, d \in j$) and by exactly three loop terms within those complex microstates.

Bimolecular Rate Model. When dealing with moves that join or break complexes, we must consider the choice of how to assign rates for each transition in a new light. In the particular situation of the join move, where two molecules in a stochastic regime collide and form a base pair, this rate is expected to be modeled by stochastic chemical kinetics.

Stochastic chemical kinetics theory [8] tells us that there should be a rate constant k such that the propensity of a particular bimolecular reaction between two species X and Y should be $k * \#X * \#Y/V$, where $\#X$ and $\#Y$ are the number of copies of X and Y in the volume V. Since our simulation considers each strand to be unique, $\#X = \#Y = 1$, and thus we see the propensity should scale as $1/V$. Recalling that $\Delta G_{volume} = RT \log \frac{V}{V_0} = RT \log \frac{1}{u}$, we see that we can obtain the $1/V$ scaling by letting the join rate be proportional to $e^{-\Delta G_{volume}/RT}$.

Thus we arrive at the following rate method, where the choice of the scalar term k_{bi} can be found by comparison to experiments measuring the hybridization rate of oligonucleotides [23], and where without loss of generality the transition from microstate i to microstate j is a join move while the transition from microstate j to microstate i is a break move:

$$k_{ij} = k_{bi} * e^{\frac{-\Delta G_{volume}}{RT}} = k_{bi} * \frac{V_0}{V} = k_{bi} * u \tag{10}$$

$$k_{ji} = k_{bi} * e^{-\frac{\Delta G_{box}(i) - \Delta G_{box}(j) + \Delta G_{volume}}{RT}} \stackrel{def}{=} k_{bi} * e^{-\frac{\Delta G_{loops}(i,j) - \Delta G_{assoc}}{RT}} \tag{11}$$

The latter simplification derives from the observation that, as in the bimolecular case, the system microstates i and j differ by exactly three loop terms in their complex microstates. However, they also differ in the total number of complexes within each system microstate, such that if i to j is a join move, $\Delta G_{box}(i) - \Delta G_{box}(j) = \Delta G_{loops}(i,j) - \Delta G_{volume} - \Delta G_{assoc}$, where $\Delta G_{loops}(i,j)$ represents the energy differences between i and j due to the three differing loop terms in the complex microstates.

This formulation is convenient for simulation, as the join rates are then independent of the resulting secondary structure. Note that an implication is that due to the rate being determined for **every** possible first base pair between two complexes, the overall rate for two complexes to bind (by a single base pair) is proportional roughly to the square of the number of exposed nucleotides (although possibly only a linear subset is likely to zipper up reliably), in addition to the $\frac{1}{V}$ dependence noted earlier.

3 The Simulator: Multistrand

Energy and kinetics models similar to these can been solved analytically; however, the standard master equation methods [24] scale with the size of the system's state space. For our DNA secondary structure state space, the size gets exponentially large as the strand length increases, so these methods become computationally prohibitive. One alternate method we can use is stochastic simulation [8], which has previously been done for single-stranded DNA and RNA folding (the **Kinfold** simulator [7]). Our stochastic simulation refines these methods for our particular energetics and kinetics models, which extends the simulator to handle systems with multiple strands and takes advantage of the localized energy model for DNA and RNA.

3.1 Data Structures

There are two main pieces that go into this new stochastic simulator. The first piece is the multiple data structures needed for the simulation: the *loop graph*, which represents the complex microstates contained within a system microstate (Fig. 3D); the *moves*, which represent transitions in our kinetics model (the single base pair changes in our structure that are the basic step in the Markov process); and the *move tree*, the container for moves that lets us efficiently store and organize them (Fig. 4).

Energy Model. Since the basic step for calculating the rate of a move involves the computation of a state's energy, we must be able to handle the energy model parameter set in a manner that simplifies this computation. Previous kinetic simulations (Kinfold) rely on the energy model we have described, though without the extension to multiple strand systems.

The energy model parameter set and calculations are implemented in a simple modular data structure that allows for both the energy computations at a local scale as we have previously mentioned, but also as a flexible subunit that can be extended to handle energy model parameter sets from different sources.

The Current State: Loop Structure. A complex microstate can be stored in many different ways, as shown in Fig. 3. While each of these has different advantages, we are going to focus on the loop representation, which allows the energy to be computed and stored in local components. One drawback is that the loop graph cannot represent pseudoknotted structures without introducing

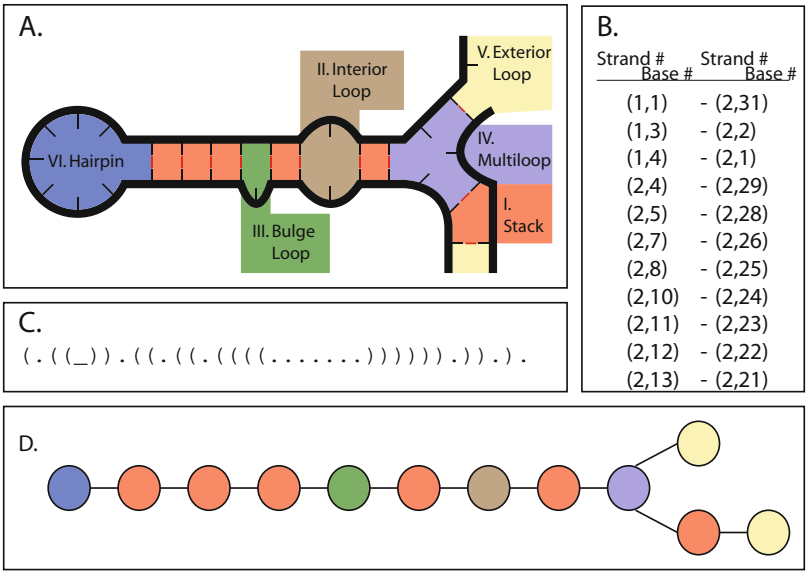


Fig. 3. Example secondary structure, with different representations: (A) Original loop diagram representation. (B) Base pair list representation. Each base pairing is represented by the indices of the bases involved. (C) Dot-paren representation, also called the flat representation. Each base is represented by either a period, representing an unpaired base, or by a parenthesis, representing a pairing with the base that has the (balanced) matching parenthesis. An underscore represents a break between multiple strands. (D) Loop graph representation. Each loop in the secondary structure is a single node in the graph, which contains the sequence information within the loop.

a loop type for pseudoknots (for which we may not know how to calculate the energy), and making the loop graph cyclic; however, since this work is primarily concerned with non-pseudoknotted structures this is only a minor point.

We use the loop graph representation for each complex within a system microstate, and organize those with a simple list. This gives us the advantage that the energy can be computed for each individual node in the graph, and since each move only affects either one or two nodes in the graph we will only have to recompute the energy for the affected nodes when performing a transition. While providing useful output of the current state then requires processing of the graph, it is a constant time operation if we store a flat representation which gets updated incrementally as each move is performed by the simulator.

Reachable States: Moves. When dealing with a flat representation or base pair list for a current state, we can simply store an available move as the indices of the bases involved in the move, as well as the rate at which the transition should occur. This approach is very straightforward to implement (as was done in the original Kinfold), and we can store all of the moves for the current state in a single global structure such as a list. However, when our current state is represented as a loop graph this simple representation can work, but does not

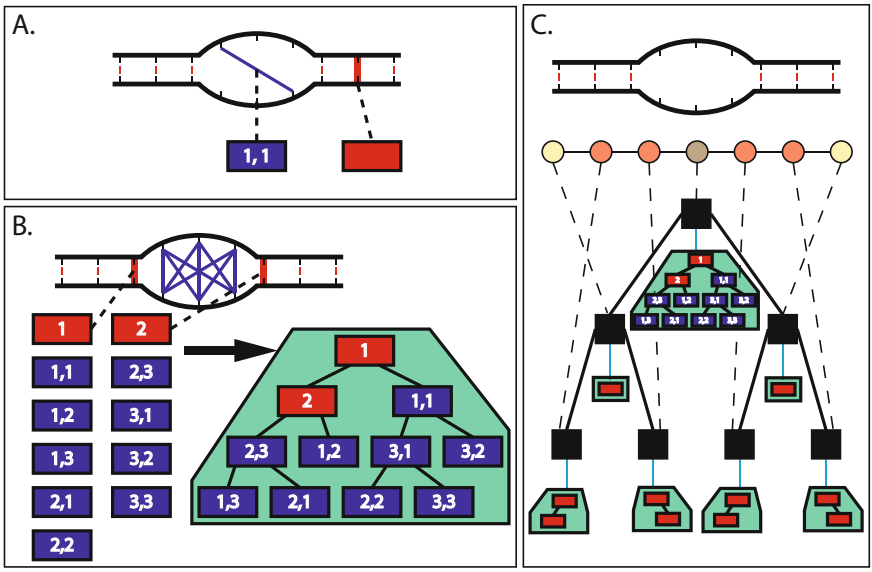


Fig. 4. (A) Creation moves (blue line) and deletion moves (red highlight) are represented here by rectangles. Either type of move is associated with a particular loop, and has indices to designate which bases within the loop are affected. (B) All possible moves which affect the interior loop in the center of the structure. These are then arranged into a tree (green area), which can be used to quickly choose a move. (C) Each loop in the loop graph then has a tree of moves that affect it, and we can arrange these into another tree (black boxes), each node of which is associated with a particular loop (dashed line) and thus a tree of moves (blue line). This resulting tree then contains all the moves available in the complex (Color figure online).

contain enough information to efficiently identify the loops affected by the move. Thus we elect to add enough complexity to how we store the moves so that we can quickly identify the affected nodes in our loop graph, which allows us to quickly identify the loops for which we need to recalculate the available moves.

We let each move contain a reference to the loop(s) it affects (Fig. 4A), as well as an index to the bases within the loop, such that we can uniquely identify the structural change that should be performed if this move is chosen. This reference allows us to quickly find the affected loop(s) once a move is chosen. We then collect all the moves which affect a particular loop and store them in a container associated with the loop (Fig. 4B). This allows us to quickly access all the moves associated with a loop whose structure is being modified by the current move. We should note that since deletion moves by nature affect the two loops adjacent to the base pair being deleted, they must necessarily show up in the available moves for either loop. This is handled by including a copy of the deletion move in each loop's moves, and halving the rate at which each occurs.

Finally, since this method of move storage is not a global structure, we add a final layer of complexity on top, so that we can easily access all the moves

available from the current state without needing to traverse the loop graph. This is as simple as storing each loop's move container in a larger structure such as a list or a tree, which represents the entire complex's available moves as shown in Fig. 4C.

3.2 Algorithms

The second main piece of the simulator is the algorithms that control the individual steps of the simulator. The algorithm implementing the Markov process simulation closely follows the Gillespie algorithm [8] in structure:

1. Initialization: Generate the initial loop graph representing the input state, and compute the possible transitions.
2. Stochastic Step: Generate random numbers to determine the next transition, as well as the time interval elapsed before the transition occurs.
3. Update: Change the current loop graph to reflect the chosen move. Recompute the available transitions from the new state. Update the current time using the time interval in the previous step.
4. Check Stopping Conditions: check if we are at some predetermined stopping condition (such as a maximum amount of simulated time) and stop if it is met. Otherwise, go back to step 2.

The striking difference between this structure and the Gillespie algorithm is the necessity of recomputing the possible transitions from the current state at every step, and the complexity of that recalculation. Since we are dealing with an exponential state space we have no hope of storing all possible transitions between any possible pair of states, and instead must look at the transitions that occur only around the current state.

4 Analysis Case Studies

We have now presented the models and algorithms that form the continuous time Markov process simulator. Now we move on to discuss the most important part of the simulator from a user's perspective: the huge volume of data produced by the simulation, and methods for processing that data into useful information for analyzing the simulated system.

How much data are we talking about here? We would expect an average of $O(N)$ moves per time unit simulated, where N is the total length over all strands in the system. This doesn't tell us much about the actual amount of data, only that we expect it to not change drastically for different size input systems. In practice this amount can be quite large, even for simple systems: for a simple 25 base hairpin sequence, it takes ∼4,000,000 Markov steps to simulate 1 s of real time. For an even larger system, such as a four-way branch migration system with 108 total bases, simulating 1 s of real time takes ∼14,000,000 Markov steps.

What can we do with all the data produced by the simulator? A key insight is that most of this Markov step data is not needed if the measurement of

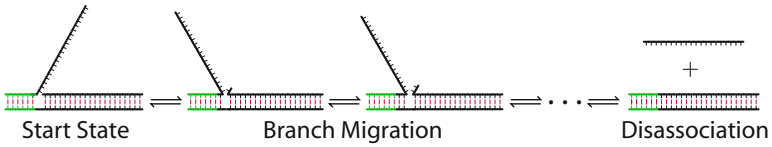


Fig. 5. Three-way branch migration system. The toehold region is in green and has sequence GTGGGT, and the branch migration region is black and has sequence ACCGCACCACGTGGGTGTCG. Both sequences are for the substrate strand.

interest is for a particular pathway, such as that shown in Fig. 5. In this system, one quantity of interest is how quickly the system reaches the completed branch migration state from the starting state. To measure this quantity we do not need to examine every Markov step as it is being made, but rather need to be able to record when we have reached the *stop state*. A stop state is typically defined by a *macrostate*, a collection of system microstates which meet some common criteria. For example, in the three way branch migration system, we might say that all system microstates which have the incumbent strand in a separate complex is the stop state of interest, as this corresponds to all the possible ways in which we could have had the incumbent strand dissociated at the end of the branch migration. For more on the definition of macrostates, see Ref. [18].

We now define the *first passage time mode* of simulation within Multistrand: given a starting system microstate and a set of stop states, it performs the simulation algorithm as given in Sect. 3.2 and records the time at which it reaches any of the stop states, as well as which one was reached. This produces a single piece of data for each trajectory simulated, which is a rather striking difference when compared to the raw number of microstates observed in a trajectory.

Let us now look at a simple three-way branch migration system in Fig. 5 and how it is to be simulated using first passage time mode. We start the system as shown, and use two different stop states: the *complete* stop condition where the incumbent strand has dissociated (as shown in the figure), and the *failed* stop condition where the invading strand has dissociated without completing the branch migration. Both of these are done using a macrostate describing a strand dissociation, which makes it very efficient to check the stop states. Note that we include the invading strand dissociating as a stop state so that if it occurs (which should be very rarely for long toehold lengths), we can find out easily without waiting until the maximum simulation time or until the strands reassociate and complete the branch migration.

The following table (Table 1) shows five trajectories' worth of data from first passage time mode on the example system. Note that we have included a third piece of data for each trajectory, which is the pseudorandom number generator seed used to simulate that trajectory. This allows us to produce the exact same trajectory again using a different simulation mode, stop states or other output conditions. For example, we might wish to run the fifth trajectory in the table again using trajectory mode, to see why it took longer than the others, or run the first trajectory to see what kinetic pathway it took to reach the failed stop condition.

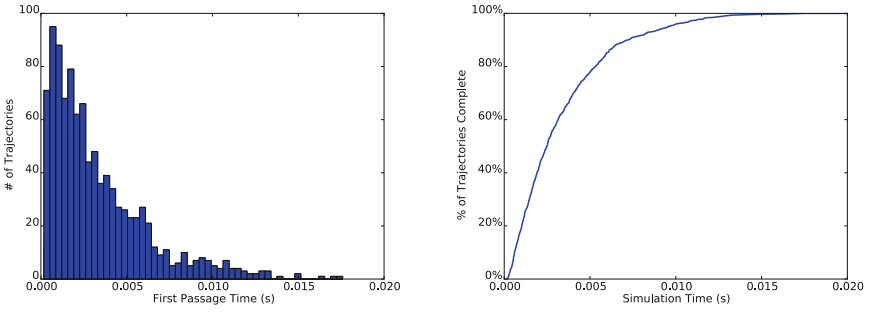
Table 1. First passage time data for the example three-way branch migration system. Stop conditions are either "complete", indicating the branch migration completed successfully, or "failed", indicating the strands fell apart before the branch migration could complete.

Random number seed	Completion time	Stop condition
`0x790e400d`	$3.7 * 10^{-3}$	Failed
`0x38188213`	$3.8 * 10^{-3}$	Complete
`0x47607ebf`	$2.1 * 10^{-3}$	Complete
`0x02efe7fa`	$2.8 * 10^{-3}$	Complete
`0x7c590233`	$6.7 * 10^{-3}$	Complete

Let's now look at a much larger data set for first passage time mode. Here we again use the three-way branch migration system shown in Fig. 5, but with a ten base toehold region with sequence GTGGGTAGGT on the substrate strand in order to minimize the number of trajectories that reach the failed stop condition. We run 1000 trajectories, using a maximum simulation time of 1s, though no trajectory actually used that much as we shall shortly see.

Instead of listing all the trajectories in a table, we graph the first passage time data for the complete stop condition in two different ways: first (Fig. 6a) we make a histogram of the distribution of first passage times for the data set, and second (Fig. 6b) we graph the percentage of trajectories in our sample that have reached the complete stop condition as a function of the simulation time.

While there are many ways to analyze these figures, we note two particular observations. Firstly, the histogram of the first passage time distribution looks suspiciously like an exponential distribution, possibly with a short delay. This is not always typical, but the shape of this histogram can be very helpful in inferring



(a) Histogram of first passage times (b) Percent completion by simulation time

Fig. 6. First passage time data for the three-way branch migration system with ten base toehold. 1000 trajectories were simulated and all of them ended with the complete stop condition.

how we might wish to model our system based on the simulation data; e.g., for this system, we might decide that this three-way branch migration process is roughly exponential (with some fitted rate parameter) and so we could model it as a one-step unimolecular process.

The second observation is that while the percentage completion graph looks very similar to an experimental fluorescence microscopy curve, they should **NOT** be assumed to be directly comparable. The main pitfall is found when comparing fluorescence curves from systems where the reactions are bimolecular: in these the concentration of the relevant molecules are changing over time, but in our stochastic simulation the bimolecular steps are at a fixed volume/concentration (reflected in the ΔG_{volume} energy term) and data is aggregated over many trajectories.

5 Conclusions

The Multistrand simulator provides a powerful platform for exploring the behaviors of molecular machines created using dynamic DNA nanotechnology. In addition to the first passage time mode described above, alternative simulation modes have been implemented to provide differing levels of detail for analysis [18]: trajectory mode provides the full elementary step trajectory, which could be used to make a movie; transition mode collects statistics on when the simulation enters and exits specified macrostates; and first step mode runs simulations starting from an initial collision, which provides an efficient method for analyzing reactions in dilute solutions. The core simulation algorithms are implemented in C++, while a flexible user interface is available from within Python. The Multistrand package can be downloaded from http://www.multistrand.org.

At this time, Multistrand is best used to explore semi-quantitative sequence-dependent phenomena, such as assessing relative sequence design quality, because kinetic predictions are not expected to be in quantitative agreement with experimental measurements. While the secondary structure energy landscape used by Multistrand agrees with established thermodynamic models such as NUPACK [4], the simple methods used to set the relative rates of different types of elementary moves (Metropolis and Kawasaki dynamics) are not flexible enough to simultaneously accurately match the widely varying rates of fundamental processes such as zipping, fraying, breathing, three-way branch migration, and four-way branch migration. This is an important area for future work.

Acknowledgements. We are greatly indebted to years of insights, suggestions, and feedback from Niles Pierce, Robert Dirks, Justin Bois, and Victor Beck, especially their contributions to the formulation of the energy model and the first step simulation mode.This work has been funded by National Science Foundation grants DMS-0506468, CCF-0832824, CCF-1213127, CCF-1317694, and the Gordon and Betty Moore Foundation through the Caltech Programmable Molecular Technology Initiative.

References

1. Crothers, D.M., Bloomfield, V.A., Tinoco Jr., I.: Nucleic Acids: Structures, Properties, and Functions. University Science Books, Sausalito (2000)
2. Chen, Y.-J., Dalchau, N., Srinivas, N., Phillips, A., Cardelli, L., Soloveichik, D., Seelig, G.: Programmable chemical controllers made from DNA. Nat. Nanotechnol. **8**(10), 755–762 (2013)
3. Chitsaz, H., Salari, R., Sahinalp, S.C., Backofen, R.: A partition function algorithm for interacting nucleic acid strands. Bioinformatics **25**(12), i365–i373 (2009)
4. Dirks, R.M., Bois, J.S., Schaeffer, J.M., Winfree, E., Pierce, N.A.: Thermodynamic analysis of interacting nucleic acid strands. SIAM Rev. **49**(1), 65–88 (2007)
5. Dirks, R.M., Lin, M., Winfree, E., Pierce, N.A.: Paradigms for computational nucleic acid design. Nucleic Acids Res. **32**(4), 1392–1403 (2004)
6. Dirks, R.M., Pierce, N.A.: Triggered amplification by hybridization chain reaction. Proc. Natl. Acad. Sci. U. S. A. **101**(43), 15275–15278 (2004)
7. Flamm, C., Fontana, W., Hofacker, I.L., Schuster, P.: RNA folding at elementary step resolution. RNA **6**, 325–338 (2000)
8. Gillespie, D.T.: Exact stochastic simulation of coupled chemical reactions. J. Phys. Chem. **81**(25), 2340–2361 (1977)
9. Hongzhou, G., Chao, J., Xiao, S.-J., Seeman, N.C.: A proximity-based programmable DNA nanoscale assembly line. Nature **465**(7295), 202–205 (2010)
10. Kawasaki, K.: Diffusion constants near the critical point for time-dependent Ising models. Phys. Rev. **145**, 224–230 (1966)
11. Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., Teller, E.: Equation of state calculations by fast computing machines. J. Chem. Phys. **21**, 1087–1092 (1953)
12. Muscat, R.A., Bath, J., Turberfield, A.J.: A programmable molecular robot. Nano Lett. **11**(3), 982–987 (2011)
13. Omabegho, T., Sha, R., Seeman, N.C.: A bipedal DNA Brownian motor with coordinated legs. Science **324**(5923), 67–71 (2009)
14. Qian, L., Winfree, E.: Scaling up digital circuit computation with DNA strand displacement cascades. Science **332**(6034), 1196–1201 (2011)
15. SantaLucia, J.: A unified view of polymer, dumbbell, and oligonucleotide DNA nearest-neighbor thermodynamics. Proc. Natl. Acad. Sci. **95**(4), 1460–1465 (1998)
16. SantaLucia, J., Allawi, H.T., Seneviratne, P.A.: Improved nearest-neighbor parameters for predicting DNA duplex stability. Biochemistry **35**(11), 3555–3562 (1996)
17. SantaLucia, J., Hicks, D.: The thermodynamics of DNA structural motifs. Ann. Rev. Biophys. Biomol. Struct. **33**(1), 415–440 (2004)
18. Schaeffer, J.M.: Stochastic simulation of the kinetics of multiple interacting nucleic acid strands. PhD thesis, California Institute of Technology (2013)
19. Seelig, G., Soloveichik, D., Zhang, D.Y., Winfree, E.: Enzyme-free nucleic acid logic circuits. Science **314**(5805), 1585–1588 (2006)
20. Shin, J.-S., Pierce, N.A.: A synthetic DNA walker for molecular transport. J. Am. Chem. Soc. **126**(35), 10834–10835 (2004)
21. Turberfield, A.J., Mitchell, J.C., Yurke, B., Mills Jr., A.P., Blakey, M.I., Simmel, F.C.: DNA fuel for free-running nanomachines. Phys. Rev. Lett. **90**(11), 118102 (2003)
22. Venkataraman, S., Dirks, R.M., Rothemund, P.W., Winfree, E., Pierce, N.A.: An autonomous polymerization motor powered by DNA hybridization. Nat. Nanotechnol. **2**(8), 490–494 (2007)

23. Wetmur, J.G.: Hybridization and renaturation kinetics of nucleic acids. Ann. Rev. Biophys. Bioeng. **5**(1), 337–361 (1976)
24. Wilkinson, D.J.: Stochastic dynamical systems. In: Stumpf, M.P., Balding, D.J., Girolami, M. (eds.) Handbook of Statistical Systems Biology, pp. 359–375. Wiley, New York (2011)
25. Xayaphoummine, A., Bucher, T., Isambert, H.: Kinefold web server for RNA/DNA folding path and structure prediction including pseudoknots and knots. Nucleic Acids Res. **33**(suppl 2), W605–W610 (2005)
26. Yin, P., Choi, H.M., Calvert, C.R., Pierce, N.A.: Programming biomolecular self-assembly pathways. Nature **451**(7176), 318–322 (2008)
27. Yurke, B., Turberfield, A.J., Mills, A.P., Simmel, F.C., Neumann, J.L.: A DNA-fuelled molecular machine made of DNA. Nature **406**(6796), 605–608 (2000)
28. Zadeh, J.N., Steenberg, C.D., Bois, J.S., Wolfe, B.R., Pierce, M.B., Khan, A.R., Dirks, R.M., Pierce, N.A.: NUPACK: analysis and design of nucleic acid systems. J. Comput. Chem. **32**(1), 170–173 (2011)
29. Zhang, D.Y., Seelig, G.: Dynamic DNA nanotechnology using strand-displacement reactions. Nature Chem. **3**(2), 103–113 (2011)
30. Zhang, D.Y., Turberfield, A.J., Yurke, B., Winfree, E.: Engineering entropy-driven reactions and networks catalyzed by DNA. Science **318**(5853), 1121–1125 (2007)
31. Zhang, W., Chen, S.-J.: RNA hairpin-folding kinetics. Proc. Natl. Acad. Sci. **99**(4), 1931–1936 (2002)