Tethered DNA hairpin structure sampling and rate of reaction

Julian Weisburd
Matt Lakin

Summer 2017

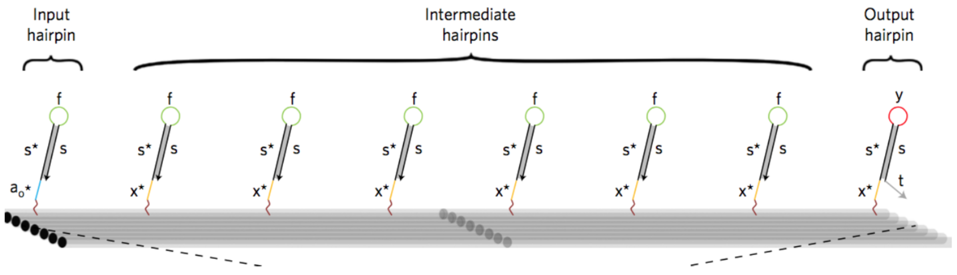

Figure 1: An example domino circuit.[1]

1 Introduction

DNA circuits can sense and respond to their environment, leading to many practical applications. A DNA circuit could take a blood sample as input, test for disease, and produce a positive or negative output. Or, DNA circuitry could provide the logic for the selective cell targeting for drug delivery.[5] DNA circuits have previously relied on diffusible components which brings some inherent limitations on the system. Diffusible components restricts scalability as it must be ensured that no cross-talk can happen between two unintended reactants.[1]

In response to the limitations of diffusible DNA circuits, "domino" circuits have been developed. In domino circuits, DNA hairpins are tethered to a DNA origami surface by a single-stranded "toehold." These tethers can bind to a diffusible input, swivel and bend around their toehold, pass their input to the next toehold, restarting the process in a domino effect until the propagated signal triggers an output (Figure 1). Because there is no risk of unintended cross-talk DNA sequences can be used multiple times to increase scalability.[1] However, like diffusible circuits, there are challenge inherent to tethered circuits. For example, The distance between two tethers? toeholds must fall within a certain range to ensure that their bound form is physically realizable. Work has been done to find this range.[4]

In diffusible circuits, the rate of reaction from input to output is easy to estimate as reaction rates are dependent on concentrations of reactants. In tethered systems, the concept of concentration becomes a little hazy. Keeping the tethers bound to a surface allows the reactive point of the DNA hairpin to move about only in a small subset of the total volume. The purpose of this project is to use *in silica* modeling to estimate the "local" or "effective" concentrations of a tethered hairpin. Doing so may provide a way to calculate the rate of reaction of a domino circuit allowing circuit designers to estimate how fast a circuit will take to produce an output in the presence of an input signal.

2 Project outline

Thus far, this project has taken form through the following steps:

1. Generate a collection of possible structures that a tethered structure could take
2. Sample from the modeled data to calculate the probability that the two tethers? reactive points colocate in 3-dimensional space
3. Calculate the local concentration from the colocating probability

3 Assumptions

Modeling inherently requires making some assumptions about the modeled system, and this section is meant to enumerate the assumptions taken during this project.

3.1 Assumptions about the DNA tether

All DNA tethers will consist of a series of domains and joints. The domains of the DNA structure are either single-stranded or double-stranded and have an integer length of nucleotides. The domains are connected by a joint which is, for the moment infinitely flexible. There are plans and the beginnings of an implementation to constrict this assumption. See the "Conclusions and Next Steps" for more details.

The length of the DNA domains in nanometers (nm) is determined by their length in nucleotides (nt). Double-stranded DNA domains have a fixed length of 0.34 nm per nt. The length of single-stranded DNA domains is variable depending on the desired distribution. In the "uniform" distribution, single-stranded DNA domains are assumed to be infinitely flexible and can therefore take on a length from 0 nm to a max length of 0.68 nm per nt. In the "worm-like chain"[9] distribution, single-stranded DNA domains are assumed to behave like a worm-like chain with all nucleotides pointing in roughly the same direction.The probability density function from used to sample single-stranded DNA lengths with the worm-like chain model is found in the supplementary info of reference [2].

Finally, the reactive points of DNA tethers are assumed to be the midpoint of the final DNA domain in the DNA structure.

3.2 Biophysical model

To model the possible physical configurations of a DNA tether, it is assumed that tethers behave like a "freely-jointed chain."[3] The domains of the DNA tether are of a fixed length (described above) and are completely independent of the orientations and positions of the neighboring domains. The orientation of each domain is determined by randomly selecting polar and azimuthal angles from a weighted distribution. This weighted distribution ensures that the orientations are uniformly sampled about a point in 3D space.[7] It should be noted that for a good portion of this project, it was assumed that randomly choosing the polar and azimuthal angles from a uniform distribution would result in a randomly chosen orientation from a uniform distribution in 3D space. As the previous reference points out, this is not the case.

3.3 Modeled system

For the remainder of this document, it will be assumed that the system attempted to be modeled is as follows. At (0,0,0) there is a toehold of a "long" tether. The long tether consists of a joint at (0,0,0), a single-stranded domain of 5 nt, a joint, a double-stranded domain of 18 nt, a joint, a double-stranded domain of 18 nt, a joint, a single-stranded domain of 6 nt, and a final joint. The "short" tether consists of a joint at (0,10.88,0), a single-stranded domain of 5 nt, a joint, a single stranded domain of 6 nt, and a final joint. The reaction between the long and short tether involves the binding of reactive point of the long domain to the reactive point of the short domain. This system is pictured in Figure 2.

It is assumed that this reaction in this system is unimolecular.[1] For a bimolecular reaction the reaction rate looks like rate $= k \cdot$ [reactant 1] $\cdot$ [reactant 2] but for a tethered reaction this becomes rate $= k \cdot$ (local concentration). The reaction is effectively unimolecular because there is no diffusion in bulk solution. However, to scale the bimolecular rate constant of the reaction to a unimolecular rate there is a need to multiply it by a local concentration.

This system was chosen because there is a literature value of local concentration of this system. From experimental data found in the supplemental pages of reference [1], the local concentration of this system is 60069.7 nM.

4 Parser

The parser is found in "string_dna_parser.py" and parses a string into a DNA tether data structure. The following string will be used as an example to describe how the simple parser works:

"$(0, 10.88, 0); J(0, \_, 0); D(0, S, 0, 1, 5); J(1, 0, \_)$"

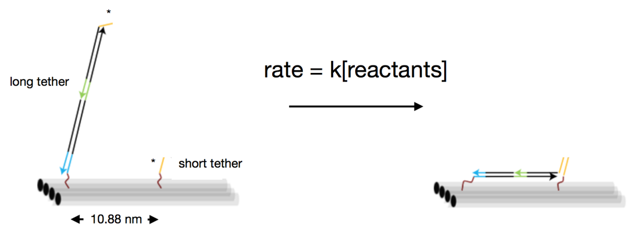

Figure 2: A cartoon representation of the two structure system modeled in this document.[1]

$(0, 10.88, 0)$
The first tuple is the toehold location of the tether in (x, y, z) coordinates.

$J(0, \_, 0)$
This tuple with a preceding "J" describes a joint.
1st position: joint ID.
2nd position: the ID of the domain in the "down" direction of the joint. "_" is a don't care character as this is the toehold joint of the structure.
3rd position: the ID of the domain the "up" direction of the joint.

$D(0, S, 0, 1, 5)$
This tuple with a preceding "D" describes a domain.
1st position: domain ID
2nd position: "Strandedness" of the domain. "S" for single-stranded and "D" for double
3rd position: the ID of the joint in the "down" direction of the domain.
4th position: the ID of the joint in the "up" position of the domain.
5th position: the length (in nt) of the domain.

5 Generating a collection of possible structures

After parsing a string, a data structure is created which contains a list of domains and a list of joints. This data structure is described in the "dna_structure.py" file. The first joint of the tether is placed at the (x, y, z) toehold position indicated in the pre-parsed string. If one wishes to move the tether, the function "move_y_tether" takes one argument and moves the tethers toehold to the passed in "y" position (in nm). It would be trivial to modify this function so it takes three arguments or a tuple to move the tether's toehold to any position in 3D space.

The algorithm to generate a collection of possible structures is simple. After reading in a joint, it finds the domain upwards from that joint. Using the strandedness, length, and, in the case of single-stranded domains, the desired distribution to sample the length from, the code generates a vector of appropriate length and random orientation (see the subsection "Assumptions?Biophysical Model" for more information about how the code samples random orientations in 3D space uniformly). This randomly generator vector is added to the (x, y, z) coordinates of the previous joint. This addition represents the (x, y, z) coordinates of the next joint. If this new joint has a negative "z" component, then it is thrown out and a new vector is randomly generated until a physically possible joint is created. This new joint is the joint read in for the next iteration of the algorithm.

The algorithm terminates after the final joint of the structure is given (x, y, z) coordinates. The reactive point in (x, y, z) coordinates of the structure is determined by finding the midpoint between the last joint and the second to last joint. This midpoint is the (x, y, z) tuple which is saved to a text file.

To generate structures, one must run the "main.py" file found in the "/src/" directory with the desired command line options enumerated in the following subsection. This creates a sister director to "/src/" called "/data/" where all the generated data are saved. The generated data are saved in files called "long.txt" or "short.txt" for the sample structures of the "long" tether and the "short" tether respectively.

At this point in time, two things are hardcoded: 1) the code generates possible values for structures that are "[5.44,10.88,12.0,13.5,21.67]" nms apart on the y axis and 2) 1,000,000 possible structures for each of the long and short structures are generated.

Options
"-w", "—worm": generates possible structures with single-stranded domains? lengths sampled from a "worm-like chain" distribution.

"-n", "—nicked": generates possible structures with angles between two double-stranded domains sampled from a "nicked" distribution. This is not implemented correctly and should not be used to generate reliable data in its current state.

If no options are specified, then the data are saved in a text file under "/data/uni_uni/⟨y_distance⟩/". If "worm" is specified, then the data are saved in a text file under "/data/worm_uni/⟨ y_distance⟩/". If "worm" and "nicked" are specified, then the data are saved in a text file under "/data/worm_nicked/⟨y_distance⟩/".

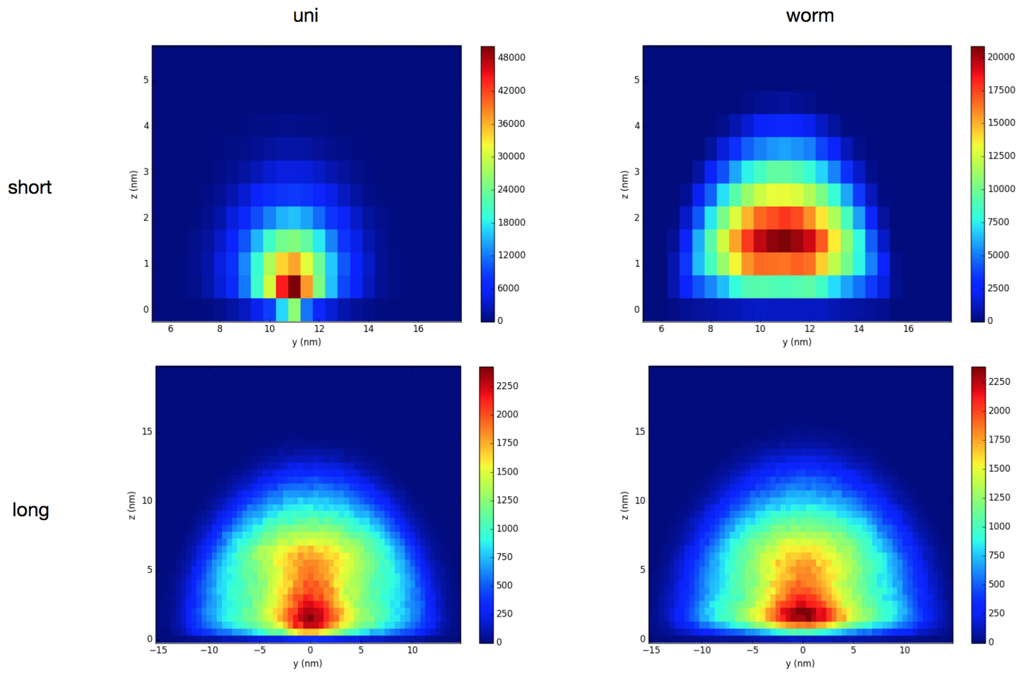

Figure 3: 2-dimensional heat maps of the reactive point of possible short and long structures generated. Two different distributions are used for sampling the length of the single-stranded domains: uniform and worm-like chain. 1,000,00 data points are used for each heat map and bin sizes are 0.5 nm.

A 2D heat map of possible structures along the (z, y) axes is show in Figure 3. It is of note how the worm-like chain distribution particularly affects the shape of the short structure, probably because the short structure is composed entirely of single-stranded domains.

6 Formula for calculating local concentrations

To calculate local concentrations, or the effective concentration that two DNA tethers may see of one another, the following formula is used[2, supplementary info]:

$$C = \frac{P}{V}$$

This formula consists of two main terms to calculate local concentration, C, in particles/nm^3 which can be derived from the generated data. P is the probability that the reactive points of the two structures colocate in 3D space. V is the reactive volume in nm^3 or the volume where two points have to coexist to be considered reacting. The next two sections of this document is dedicated to describing the various methods used to calculate these two terms and local concentrations.

Additionally, the unit conversion used in this work are provided below. This is where the magic number 1660577881 in the file "/analysis/constants.py" comes from.

$$\frac{1 \text{ mol}}{6.022 \cdot 10^{23} \text{ particles}} \frac{1 \cdot 10^{27} \text{ nm}^3}{1 \text{ m}^3} \frac{1 \text{ m}^3}{1000 \text{ L}} \frac{1 \cdot 10^{9} \text{ nM}}{1\text{M}}$$

7 Calculating colocation probabilities

Calculating the probability that the reactive point of the "long" structure colocates with the reactive point of the "short" structure (or vis-versa) has taken two forms: binning and thresholding. As described in the subsequent section, binning isn't viable to accurately calculate local concentration although it is included here because its concepts are used to cut down processes time.

Binning
Binning the data involves reading in every point from the generated sample structures and categorizing them into discrete bins. In the code, this is done by setting a bin width and rounding the point to the nearest multiple of that bin width. For example, if a bin width of 0.5 nm is set, a point with the coordinates (17.89,4.378,0.123) would be binned to (18.0,4.5,0.0). The data is read into two maps, a short map and a long map with the rounded bins acting as the keys and exact points as the values. After the data is read into the two maps, a probability of colocation is calculated by counting the number of short samples in a bin, dividing by the number of total short samples and the multiplying this value by the number of long samples in the same bin and dividing by the total number of long samples. This probability is then calculated for every bin in the two maps and summed up to result in a total probability that the short and long structures? reactive points are colocating in 3D space space.

Thresholding
Thresholding the data involves reading in every point from the generated sample structures and then setting a threshold distance which serves as the maximum distance a long reactive point can be from a short reactive point for the two to be considered colocating in space. To calculate the probability, the code first bins all the data as described above.

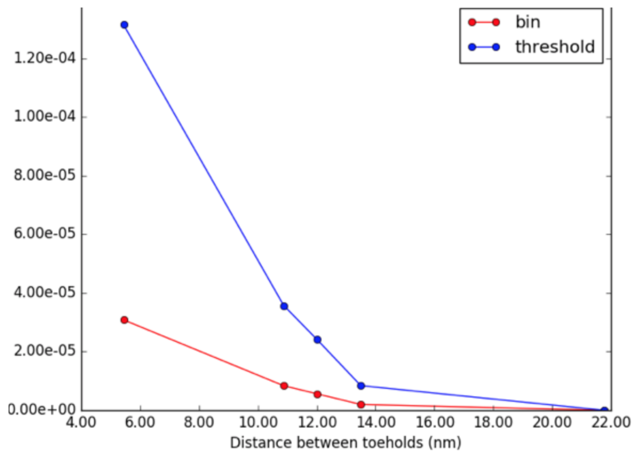

Figure 4: Probability of short and long structures colocating as distance between the two structures increases. Comparison of binning and thresholding techniques. Bin/threshold size 0.5 nm

It then iterates through all the short point data and looks for long points within the threshold distance in the 27 bins around it. The probability of colocation is then the number of long points within a threshold distance of that short point divided by the the total number of long point samples.

A comparison of the probabilities derived from the two procedures is shown in Figure 4.

8 Calculating local concentration

Originally, Local concentrations for the system were found by computing the colocation probabilities for the whole system and then dividing by an estimate of the reactive volume for the whole system as well. That is, in the case of binning, the probabilities found of colocation for every bin were added together to represent a total probability. In the case of thresholding, every short structure point's probability of colocation was supped up and averaged. This value was then divided by a reactive volume.

The reactive volume was a little ambiguous as a concept. In its earliest form, it represented the total reactive volume of the whole system. For binning, the reactive volume was the total volumetric sum of all the bins where a long reactive point and a short reactive point were colocating. This gave some relatively large reactive volumes which resulted in some really small computed local concentration values. Another way the reactive volume of the whole system was represented was just the volume reachable

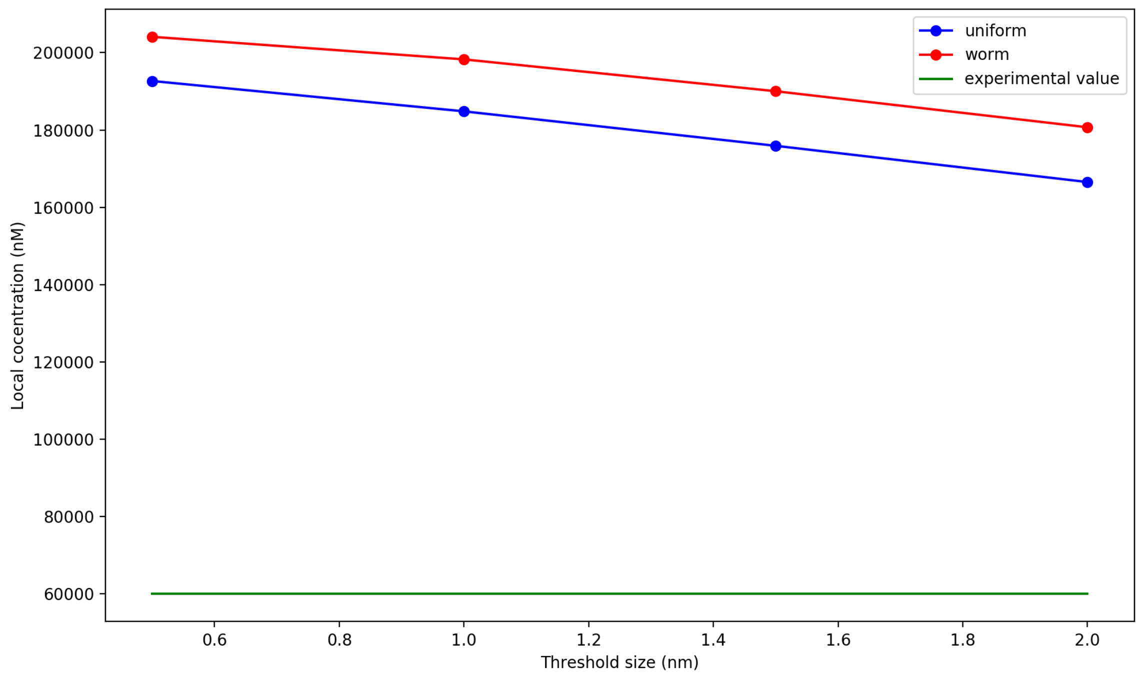

Figure 5: Computed local concentration of modeled system. Threshold size increases along the x-axis. The experimental value of 60069.7 nM is shown as the constant green line.

by the short structure?s reactive point. This was a hemisphere above the x, y plane and, again, was a relatively large reactive volume.

Currently, local concentrations are calculated by considering the local concentration at each point of the short structure. These local concentrations are found by calculating the probability of colocation at that short structure's point through thresholding and then dividing by the reactive volume, or the sphere around the short structure's point using the thresholding size as a radius. These local concentrations are then averaged to give a local concentration of the whole system. The results of this procedure are shown in Figure 5. A 2D representation of the local concentration at each short structure point is shown in Figure 6.

To calculate local concentration, one must run the file "/analysis/local_concentrations.py". This file will save the computed local concentrations using a distribution combination

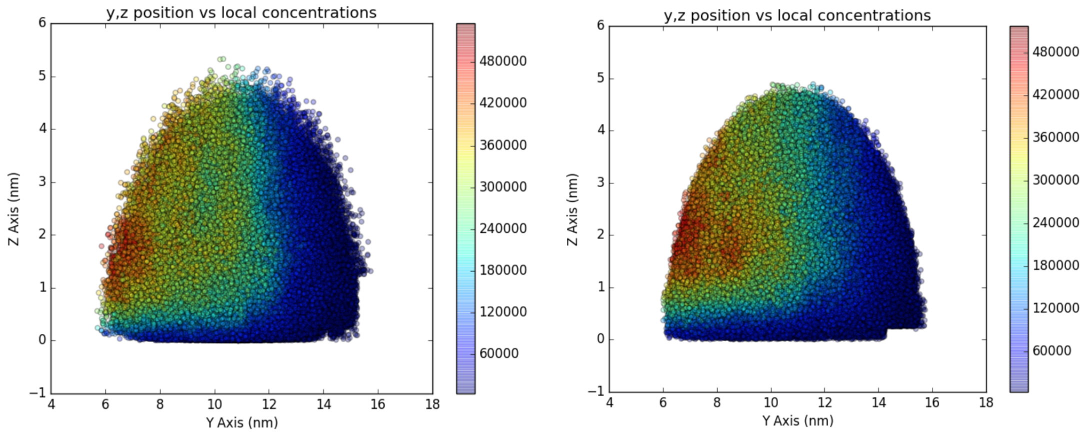

Figure 6: A y, z scatter plot of local concentrations computed at sampled short structure reactive points. "Hotter" color corresponds to a higher local concentration. Single-stranded domain length drawn from a uniform distribution is shown on the right and worm-like chain distribution is shown on the left.

"uni_uni" and "worm_uni" and save the results to a text file. Depending on the amount of data from the structure generation step described in a previous section, this process may take a long time.

9 Conclusions and next steps

As seen in Figure 5, the calculated local concentrations are about three times greater than the experimental value of 60069.7 nM. This seems to be a decent first attempt at calculating local concentration as the results were within a relatively small factor of the experimental value. This section lists some possible steps to take in the future in order to improve the calculated value.

There are a number of biophysical models which may more accurately describe a tethered DNA hairpin structure. For example, the aforementioned worm-like chain could be used for not only a distribution of lengths of single-stranded DNA but for modeling the whole structure.[6] A Markov chain Monte Carlo also has been used in the literature to model a tethered chain.[8]

A further improvement involves implement angle constrictions between two double-stranded DNA domains instead of allowing a infinitely flexible joint between them. Doing so requires sampling from distribution of possible angles between the two domains. This distribution is taken from the supplementary information of reference 7, and the raw data was kindly supplied to us from the authors of that paper. The file "nicked_distribution.py" is responsible for reading in the raw data and sampling from an estimated probability density function derived from those data. Specifically, after initializing a "NickedDistribution" object, calling "generate_rand_from_pdf" on that object will return a single value from that distribution.

10 References

[1] Chatterjee, Gourab, et al. "A Spatially Localized Architecture for Fast and Modular DNA Computing." Nature Nanotechnology, vol. 12, July 2017, pp. 920-927.

[2] Genot, Anthony, et al. "Remote Toehold: a Mechanism for Flexible Control of DNA Hybridization Kinetics." Journal of the American Chemical Society, vol. 133, no. 7, Jan. 2011, pp. 2177-2182.

[3] "Ideal Chain." Wikipedia, en.wikipedia.org/wiki/Ideal_chain.

[4] Lakin, Matthew, and Andrew Phillips. "Automated, Contstraint-Based Analysis of Tethered DNA Nanostructures." DNA Computing and Molecular Programming, pp. 1-16.

[5] Muscat, Richard, et al. "DNA-Based Molecular Architecture with Spatially Localized Components." ACM SIGARCH Computer Architecture News, vol. 41, no. 3, June 2013, pp. 177-188.

[6] Song, Dan, et al. "Tethered Particle Motion with Single DNA Molecules." American Journal of Physics, vol. 83, May 2015, pp. 418-426.

[7] "Sphere Point Picking." Wolfram MathWorld, mathworld.wolfram.com/SpherePointPicking.html.

[8] Van der Heijden, Thijn, and Cees Dekker. "Monte Carlo Simulations of Protein Assembly, Disassembly, and Linear Motion on DNA." Biophysical Journal, vol. 95, no. 10, 15 Nov. 2008, pp. 4560-4569.

[9] "Worm-like Chain." Wikipedia, en.wikipedia.org/wiki/Worm-like_chain.