8-2023

Simulation and Analysis of Self-Assembling Slat-Based DNA Ribbons

Lukas Vaughan

Hunter J. Fleming
University of Arkansas, Fayetteville

Simulation and Analysis of Self-Assembling Slat-Based DNA Ribbons

Simulation and Analysis of Self-Assembling Slat-Based DNA Ribbons

A thesis submitted in partial fulfillment
of the requirements for the award of
Honors for Bachelor of Science in Computer Science

By

Hunter Fleming and Lukas Vaughan
University of Arkansas

April 2023
University of Arkansas

This thesis is approved for recommendation to the Honors College.

Prof. Matthew Patitz
Honors Advisor and Committee Member

Prof. Susan Gauch
Committee Member

Prof. John Gauch
Committee Member

Prof. Thi Hoang Ngan Le
Committee Member

Prof. Ukash Nakarmi
Committee Member

Abstract

Though still in its infancy, the design of DNA crisscross slats presents great potential in the algorithmic self-assembly of DNA. The provision for higher levels of cooperativity allows for fewer errors through the natural proofreading of slat placement, leading to more robust assembly. Highly accurate simulations of self-assembling DNA squares have been achieved by following the kinetic Tile Assembly Model. Building on that foundation, this study seeks to calibrate the system parameters of a kinetic simulator for self-assembling DNA slats to match experimental results and to use those ranges of parameters to perform exploratory simulations of systems not yet tested in a lab setting. Novel systems include those with fewer unique slat types to analyze the trade-off between growth rate and accuracy of each assembly.

THESIS DUPLICATION RELEASE

I hereby authorize the University of Arkansas Libraries to duplicate this thesis when needed for research and/or scholarship.

Agreed_____

Hunter Fleming

Refused_____

Hunter Fleming

Agreed_____

Lukas Vaughan

Refused_____

Lukas Vaughan

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 Introduction

1.1 Problem Statement

In the field of DNA tile-based self-assembly, kinetic models are simulation models that seek to capture elements of real-world chemistry and thermodynamics that are either ignored or oversimplified by abstract modeling. Kinetic models represent real-world properties such as tile concentration and temperature in the form of free energy variables G_{mc} and G_{se}, as well as k_f, or forward rate constant, which is representative of units of time [1]. These parameters generally have ranges in which accurate, controlled growth is observable that corresponds to real-world experimental setups.

DNA polyomino slats are an alternative structural approach to the construction of DNA tiles for self-assembly, as opposed to traditional square tiles [2]. Slats consists of rods subdivided into singular binding domains and are formulated with the idea that they allow for much better use of cooperative growth than traditional square tiles and thus allow for much greater error checking. These slats are arranged into macrotiles with assemblies varying in cooperativity, referring to the number of attachments needed for a slat to be stable, and slat count, referring to the uniqueness of slats in a given macrotile. Differences have been observed in the apparent relationship between kinetic model parameters and real-world experimental parameters under polyomino slats versus square tiles [2].

The purpose of this project was to investigate the parameters utilized in kinetic modeling to find ranges in which accurate, controlled growth is possible in systems that use DNA polyomino slats. This consisted of thousands of runs of kinetic simulations in an effort to produce results that resembled those found in [2], with the ultimate aim of investigating the variance of these parameters at different levels of cooperativity and different slat counts and, if necessary, reformu-

lating known formulae and generalizations already established in square tile kinetic modeling for polyomino slats.

1.2 Background and Definitions

1.2.1 Algorithmic Self-Assembly of DNA and the aTAM

The mathematics of algorithmic self-assembly can be applied to DNA structures to create systems that spontaneously assemble according to logical rules. A visualized system in the abstract Tile Assembly Model (aTAM) [1] consists of square tiles with "glue" labels and strength values on each side. A tile attaches to another tile or group of tiles if the corresponding glues have matching labels and if the combined strength of the resulting bonds is greater than or equal to the model variable "system temperature." Temperature in the aTAM designates the minimum combined binding strength required for tile attachment [1]. Cooperative growth, the idea that multiple sides of a tile must be paired with their respective counterparts to successfully attach given a system temperature, enables algorithmic growth through careful manipulation of glue labels and strength values [1].

As this is self-assembly, the attachment of tiles must be a spontaneous process. While the aTAM is Turing complete [3], this only occurs at a minimum temperature of 2 due to the requirement of cooperative growth. Cooperative growth, however, relies heavily on proper orientation and positioning of tiles in an assembly, which take prohibitive amounts of time to assemble entirely on their own. Seed tiles or seed assemblies are used to facilitate this type of growth by providing a base superstructure for additional tiles to attach to that theoretically only allows tiles to attach already in the proper orientation and position in the assembly. DNA can implement the aTAM through a design motif that uses a self-assembled square of DNA with single strands of nucleotides extending from each side of the square to act as the glues [1]. Careful design of those sequences results in their use as glue labels that attach to their complementary sequence on another

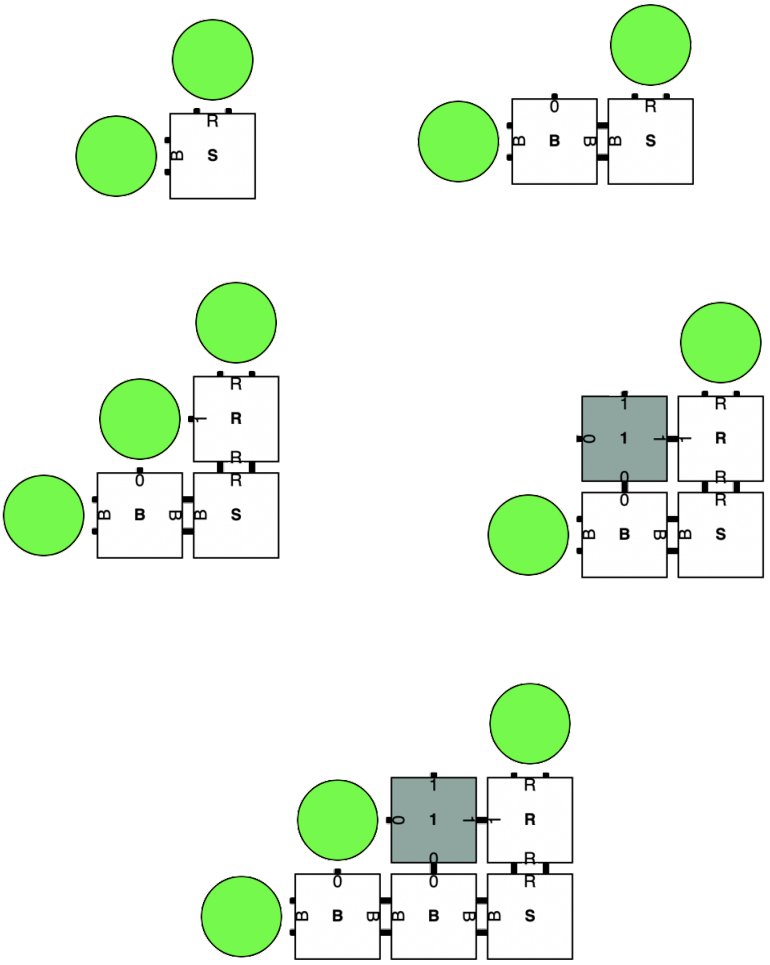

Figure 1.1: Example of a system in the aTAM assembling under the constraint of cooperative growth.

tile. Ultimately, these DNA tiles can self-assemble into structures [4] and perform computation [1].

1.2.2 Kinetic Tile Assembly Model (kTAM)

In the aTAM, tiles that bind together do not come unbound, and no tile binds with mismatched glue labels or with insufficient glue strength. In the reality of chemistry and the associated thermodynamics, tiles binding correctly and continuing to stay bound is only statistically true. Even strongly bound tiles can detach occasionally, and the stability of tile attachment is influenced by factors such as the physical temperature of the solution and the relative and absolute concentration of tiles. Likewise, tiles not designed to bind in a specific location will often do so for a short amount of time, likely to detach again afterward. However, if other tiles attach after an erroneous tile, the error can become trapped inside the structure and likely propagate additional erroneous growth through additional tiles attachment. The mechanisms influencing this dynamic attaching and detaching of DNA structures can be described by the kinetic Tile Assembly Model (kTAM) [1]. Under this model, temperature, concentration, and the length of glue strands can be varied to influence the dynamics of assembly. Higher physical temperature results in increased energy in the system and makes tiles more likely to dissociate from others. Lower temperature allows for easier bonding for all tiles, incorrect pairings included. This can lead to so-called "tumorous" growth, or more formally unordered growth, where tiles attach correctly or incorrectly regardless of the intended algorithm. The most accurate growth occurs at the slowest growth rates associated with temperatures just below the melting point. At this temperature, incorrect attachments are highly likely to dissociate before becoming trapped, and only correct tiles are likely to remain for meaningful amounts of time. Though influencing the forward rate in a similar way to temperature, the concentration of the constituent tiles plays a much larger role than temperature in influencing how often those tiles randomly collide. An increase in concentration increases the colli-

sion frequency of both correct and incorrect pairings, but whether those collisions result in double helix bonds and whether those bonds remain for a meaningful amount of time are dependent on the lengths and sequences of the oligonucleotide "glues" and temperature [1]. The length of oligonucleotides, if relatively small, is assumed to have negligible affect on the rate of collisions between tiles and is not a factor in the forward rate equation of the kTAM [1]. Concentration is assumed to have no impact on how often bonds break and is not a factor in the dissociation equation of bonds. Three parameters of the kTAM express the effect of temperature, concentration, and glue length on the association and dissociation of tiles:

$$G_{mc},\ G_{se},\ k_f$$

G_{mc} generally describes the free energy cost of positioning a tile for attachment and is highly dependent on tile concentration as well as a term α that describes the entropic loss of attaching to the assembly among other factors. Higher values of G_{mc} make it harder for tiles to attach to an assembly while lower values make it easier. G_{se} describes the free energy cost of singular bond dissociation [1] [5] and is linearly proportional to the free energy stored in a single bond as well as inversely proportional to the temperature of the system. k_f describes the time scale of simulation and is an experimentally determined constant. It is related to physical time and is influenced by concentration, temperature, and oligonucleotide length. Tile systems have been simulated with a wide range of these parameters to map out which combinations give no growth due to a prohibitively high cost of association or too cheap a cost of dissociation, tumorous growth due to an unnaturally low cost of association or a high cost of dissociation, and a region of ideal growth where the formation of desired seed-initiated, error-free aggregates occurs [1] [5]. The specific kTAM equations [5] behind each of the three parameters listed are:

$$G_{mc} = \alpha - \ln(\frac{[c]}{u_0}) \qquad\qquad G_{se} = \frac{-\Delta H^\circ + T\Delta S^\circ}{RT}$$

$$\hat{k_f} = u_0 k_f e^{\alpha}$$

Both G_{mc} and G_{se} are related to the free energy of bonds. α is "a constant unitless free energy change from other factors, such as the loss of rotational entropy during binding" [5]. $[c]$ is the concentration of tiles, and u_0 is a constant used for unit conversion. ΔH° and ΔS° represent the change in enthalpy and entropy respectively from the standard definition of Gibbs free energy and for G_{se} describe the thermodynamics of dissociation of bonds. T is temperature, and R is the universal gas constant. The k_f referred to throughout this thesis is actually $\hat{k_f}$, the effective rate constant, which is dependent on an actual forward rate constant called k_f which has been experimentally calculated to be around 10^6 for the crisscross slat ribbons used in [2], but this constant is highly dependent on the kind of tiles used and how they are implemented in DNA. The forward and reverse rates of a kTAM system, r_f and $r_{r,b}$, are described by the following equations in terms of the previously defined parameters and b, "the total strength of the correct bonds between the tile and adjoining tiles" [1] [5]:

$$r_f = \hat{k_f} e^{-G_{mc}} \qquad\qquad r_{r,b} = \hat{k_f} e^{-bG_{se}}$$

It has been shown that the ratio of G_{mc} to G_{se} roughly corresponds to the system temperature from the aTAM [1] [5]. The melting point, the point at which the cost of tile attachment is too great relative to a tile's ability to remain bound resulting in prohibitively difficult growth, is defined as the point at which this ratio exceeds cooperativity. When the ratio of G_{mc} to G_{se} is less than about half the cooperativity, defined as the precipitation point, the cost of bond dissociation is too high for accumulated errors to detach, leading to inaccurate growth. Only between approximately half the cooperativity and the cooperativity does the ratio of parameters generally produce measurable, accurate growth.

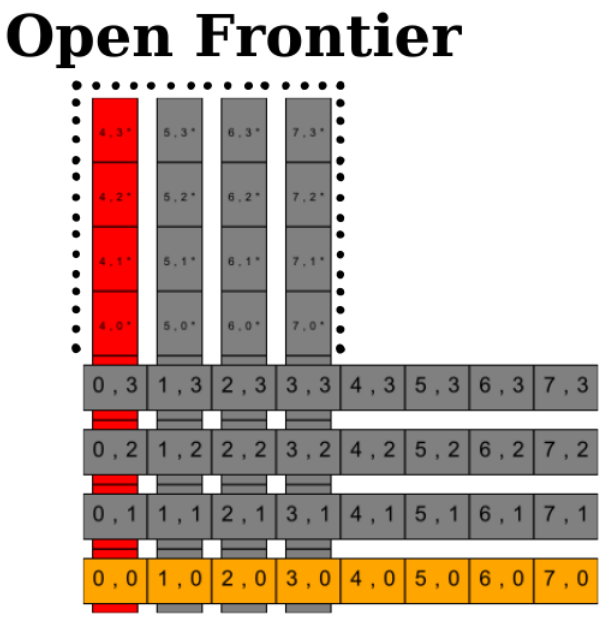

Figure 1.2: Slats composing a macrotile

1.2.3 DNA Slats

DNA origami slats are an approach to polyominos, a generalization of the logic of square tiles, that envisions functional or logical "macrotiles" constructed of individual slats bound together [2] (Figure 1.2).

Slats attach to a superstructure, and when enough have arranged themselves in the proper geometry, additional slats can begin to bind on the newly exposed frontier. This is a major difference from square tiles, whose individual attachments each immediately expose available frontier for subsequent attachments. While traditional square tiles are limited to cooperative growth requiring two or three other tiles to be in place, slats have no theoretical limit to the amount of

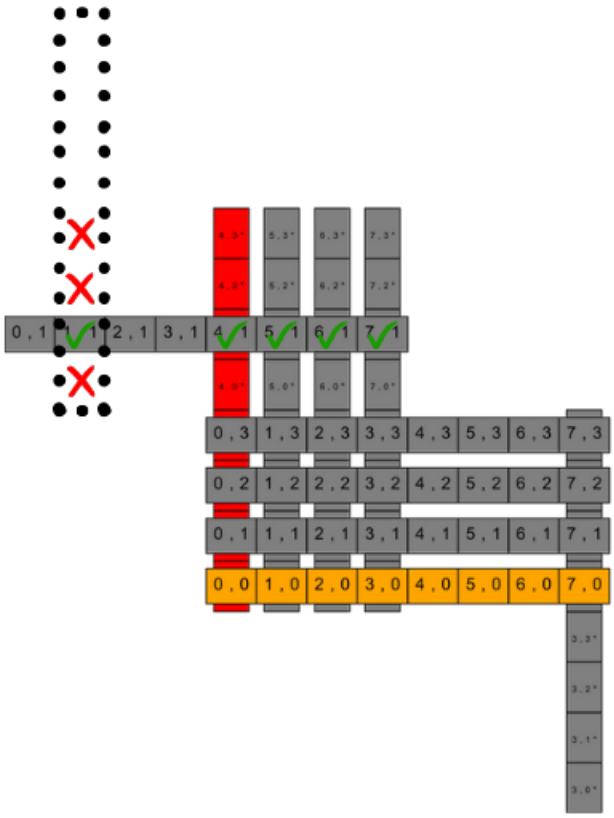

Figure 1.3: Cooperativity of a slat system visualized.

possible cooperativity. The limit of cooperativity is only bounded by the manufacturing limitations associated with designing ever-longer slat motifs. Slats bind together into units generally referred to as macrotiles, which can be thought of as a basic logical or functional unit of an assembly while individual slats comprise the basic structural unit of an assembly. A macrotile consists of multiple slats, the arrangement of which will allow other slats to attach and form the next macrotile in the superstructure. In theory, this additional cooperativity should allow much stronger error correction capability than that afforded by traditional square tiles, as slat cooperativity and reliance on logical macrotiles forces multiple errors to accumulate before such errors become locked in place and trapped in the superstructure. Hypothetically, for cooperativity N, the only way for an erroneous macrotile to propagate is if N matching, erroneous slats remain attached simultaneously. Because erroneous slats are more likely to detach than remain, the chance of N matching, erroneous slats remaining attached long enough for a macrotile to propagate becomes exponentially low with increasing levels of cooperativity. The limits on slats in real-world experimentation are largely limits of cost and time, while the main limiting factor in simulation is computational complexity. Thus a core goal of this project was to evaluate a kinetic model designed for use with slat tiles as well as develop methodology for conducting kinetic simulations in a timely manner, with the ultimate purpose of further evaluating the potential of slat tiles for use in algorithmic self-assembly.

1.2.4 RodSim

Just as the kTAM and kTAM simulators, such as xgrow [6] and ISU TAS [7], have allowed for accurate simulation of systems not yet physically designed, a mathematical model and computer program capable of simulating the kinetics of slat self-assembly are highly desirable. Building on the assumption that the kTAM equations or variations of them could describe the kinetics of slat self-assembly, a simulator called RodSim has been created by Dr. Matthew Patitz

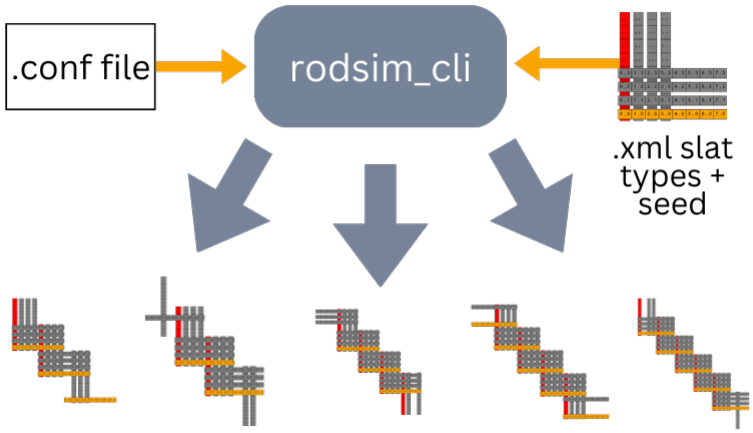

Figure 1.4: Functionality of RodSim: The configuration file and the starting .xml file containing the slat types, the seed assembly, and other system parameters are fed into rodsim_cli, which outputs the current assembly in regular intervals throughout the simulation.

and Daniel Hader to attempt to model this. An xml file containing a description of the system to be simulated is fed into the program. The xml file contains the description of unique slat types, the cooperativity of the system, the G_{mc}, G_{se}, and k_f parameters, and the starting seed assembly. During program execution, the current step of the assembly process can be output in regular intervals to xml files which can be interpreted as images using other software. It is not known, however, if the ratios of G_{mc} and G_{se} from the kTAM hold for the slat model that RodSim is built on.

1.2.5 Slats and Kinetics Terminology

This paper will use a few terms to describe slats, here defined. Firstly, **cooperativity** refers to the number of binding domains a slat must attach to to be considered stable and correct growth. A cooperativity of 16 means that a

slat attaches to 16 other slats, and itself most likely propagates 16 open domains for other slats to bind to. Slats can generally only arrange themselves such that vertically-oriented slats bind only to horizontally-oriented slats and vice versa. Secondly, **slat count** refers to the number of vertical and horizontal slats that are considered unique. A **binding domain** consists of a nucleotide glue onto which a complementary glue binds. Slat count most specifically refers to the uniqueness of slats within a given macrotile, and can thus be thought of as specifying the range of possible errors in a macrotile or assembly. This term is generally defined relative to cooperativity, so for a cooperativity of 16 (denoted v16), a slat count of 2 means that the slats require 16 binding domains to attach and there are two unique slat types comprising a full macrotile, that is a full horizontal or vertical stretch. Further subdivision consists of auto-correlation, or self-correlation, and cross-correlation. **Auto-correlation** refers to the maximum number of bonds with which a slat of a given direction can bond incorrectly, while **cross-correlation** refers to the maximum number of bonds with which a slat of a given direction can bond with a slat of the opposite direction. While cross-correlation is not handled by the simulator used for this paper, auto-correlation corresponds to cooperativity and slat count as $\frac{cooperativity}{slatcount}$; thus, a system with cooperativity 16 and slat count of 2 would have an auto-correlation of 8.

In reference to kinetic simulation, there are three parameters of interest:

- G_{mc} - a free energy representation of primarily tile concentration that is defined generally as the amount of energy required for a tile to attach to an assembly.

- G_{se} - a free energy representation of primarily temperature that is defined generally as the amount of energy required to break a 1-strength bond between tile glues.

- k_f - an abstract constant representative of the units of a simulation's time axis.

2 Methodology

2.1 Approach

Starting from the equations defined for the kTAM in an earlier section, we test a hypothesis made by Dr. Matthew Patitz's lab group that the only difference between those equations and ones applying to a system of slats is the value α. This variable, partially describing "the loss of rotational entropy during binding," [5] is thought to be much higher for systems of slats, where the unwieldy length and additional binding domains make alignment to a superstructure slower.

To generate a broad map of how the three parameters G_{mc}, G_{se}, and k_f affect the assembly of slat systems, and to narrow the ranges of parameters to those that facilitate realistic, accurate growth, a myriad of simulations run at different combinations of parameters is needed. We describe a software pipeline that can be used to facilitate these simulations and analyze the results. By "realistic" growth, we refer to assemblies which have been tested in a lab setting and have experimental evidence to validate RodSim and a given choice of parameters. As such, for the purpose of tuning a simulator to match experimental outcomes, we were generally not looking for the values of G_{mc}, G_{se}, and k_f that produced the best results. Instead, we generally looked for parameters that produced results most closely resembling data found in physical experimentation.

2.2 Pipeline Architecture

2.3 Description of Software

The following is a description of the software pipeline used to conduct simulations, along with various utility scripts used for organizational or other purposes.

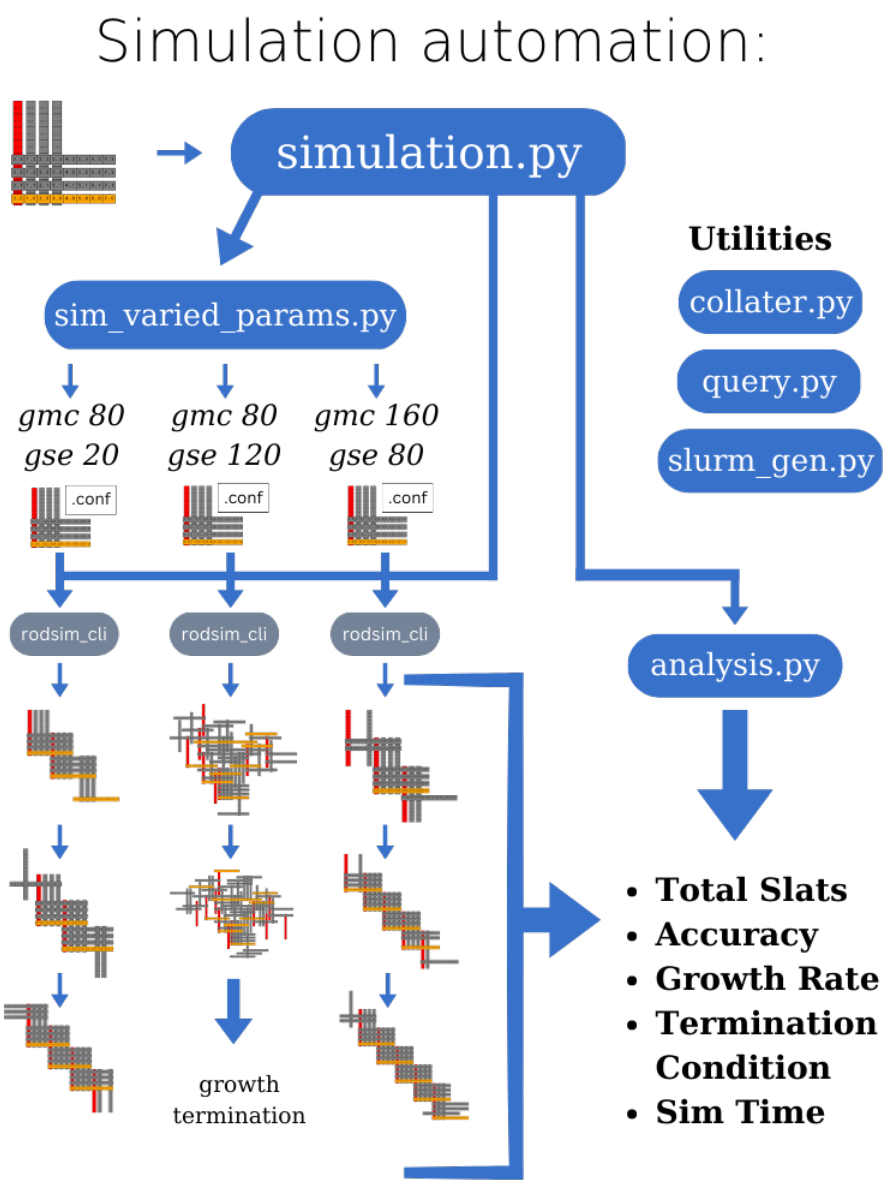

Figure 2.1: High-level architecture of the Python scripts used to automate simulations on the server.

Main Pipeline

rodsim_cli - C++ kinetic simulator written by Daniel Hader of Dr. Patitz's research group. Operates on configuration files specifying simulator arguments along with xml files describing tile systems. Xml files specify one set of G_{mc}, G_{se}, and k_f, and thus one xml file and one configuration file is needed for each set of parameters tested. The simulator outputs xml files at preset intervals and can also be run in an abstract mode for aTAM simulation.

simulation.py - the primary controlling Python script for simulations. Takes a range of each parameter of interest to be run on a given assembly, along with varying arguments for the other programs in the pipeline. For the purposes of parallelization, this was the root script from which any simulation was launched. It has three primary modes of operation: Full, Partial, and Analyze. Full means a full instance of the simulation pipeline, with input file variation, simulation, and analysis steps. Partial includes only the simulation step and is designed to run on a list of configuration files. Analyze runs only the analysis step and is designed to run on a directory containing simulation results.

sim_varied_params.py - Python script designed to be executed by simulation.py as the variation step of the pipeline. Takes an input xml file and inclusive range of parameters and generates input files and file structure for the simulation step.

analysis.py - Python script designed to be exectued by simulation.py as the analysis step of the pipeline. Takes two xml files, one input and one 'golden' key file, as inputs and produces separate csv files for each set of parameters. This script also calculates the accuracy and growth rate of the simulation, as well as identifying some notable features in the assembly such as trapped slats or premature slats.

Utility Scripts

exponential_kf.py - Allows for a set of systems to be generated over non-linear increments of parameters.

slurm_gen.py - Additional grandparent script used only for server purposes in generating the Slurm scripts that submit batches to the queue. This script also serves as the outermost layer of work distribution for parallelization, as it controlled

dividing a set of parameters into smaller chunks to be distributed amongst Slurm jobs. Takes the same arguments as simulation.py along with some Slurm-specific arguments.

collater.py - Recursive csv merger. Takes a target directory and concatenates every csv below that directory. Used for gathering results from individual csv files generated by the simulation pipeline.

marker.py - Script for marking incomplete simulations from a batch of simulations that timed out or otherwise failed. Produces a list of configuration files that can be used as input by the Partial mode of simulation.py. Determines completion of a simulation by the presence of final output files (output files that have reached the maximum number of steps for the simulation) or of csv files.

query.py - Retrieves information for specific simulation runs matching parameter criteria, exactly or in a range, writing the new results to a .csv if a flag is asserted; retrieves the final .xml assemblies of the runs requested and groups those into an output directory also.

matplot_viewer.py - Basic framework for viewing results using matplot 3d graphs. Set up for using filters on dataset and reconfigurable axes, colors points based on accuracy and optionally also growth rate.

3D_plotter.py (created by Dr. Patitz) - More powerful 3D viewing of systems; less lag, better visualization, and more freedom to move, zoom, and view individual simulation run values; both this and the previous script can be used to identify "shells" or regions with correct growth at desired growth rates.

2.4 Pipeline Details

The general pipeline used for simulation was a multithreaded process designed to run hundreds of individual simulations at once on a computing cluster. This project utilized resources from the AHPCC (Arkansas High Performance Computing Center) to accomplish this. As individual simulations can take hours to complete to any significant depth, large-scale parallelization of the process was

the best option for producing a large volume of data to work off of in a reasonable amount of time.

The pipeline's general input consisted of a set of xml files, one input file that serves as the base assembly and one 'golden' or key file that serves as an example of correct growth for the purpose of comparison. These golden files were generated using the same input assembly and rodsim_cli executed in abstract modeling mode. The only other strictly required input was a range of values for each of G_{mc}, G_{se}, and k_f respectively to be tested, with the pipeline generating all necessary rodsim_cli configuration files and the necessary input files.

The pipeline's output would be periodic printouts of the assembly at regular step intervals, these xml files being produced by rodsim_cli itself. Additionally, when all simulations have completed, the pipeline's analysis step would then produces separate csv files for each individual set of parameters, these csvs to be collated together after program execution and outside the main pipeline. Csvs store a variety of information about the assemblies. The features stored consisted of:

- Process - this is essentially the name of the simulation that produced this specific point, stored as the path to the source xml file.
- Key - this is the name of the golden file used for comparison.
- Assembly - this is a general name given to the assembly being tested.
- Steps - this is the number of steps that had been simulated at the time the source xml file was outputted.
- Min_Bind - this is the minimum binding threshold needed for rodsim_cli to consider possible attachment to a superstructure. Set as a command line argument.
- Max_Attachments - this is the maximum number of attachments to the superstructure rodsim_cli will allow before terminating. Set as a command line

argument.

- Time - this is the elapsed time calculated by rodsim_cli that estimates the amount of real time that would have passed in a physical experiment.
- G_{mc} - this is the value of G_{mc} used for the simulation that produced this output file.
- G_{se} - this is the value of G_{se} used for the simulation that produced this output file.
- k_f this is the value of of k_f used for the simulation that produced this output file.
- Total Slats - the total number of slats, including the seed, attached to the assembly.
- Correct - the number of slats in the assembly considered to be correct, excluding the seed. This is calculated by comparing the position and identity of slats between the output file and the golden file.
- Incorrect - the number of slats in the assembly considered to be incorrect. This is calculated as the Total Slats - seedSlats - correctSlats.
- Trapped - the number of slats in the assembly considered to be trapped. A trapped slat is defined as a slat which is both incorrect and which has a binding strength greater than or equal to $\frac{G_{mc}}{G_{se}}$.
- Premature - the number of slats in the assembly considered to be premature. A premature slat is defined as a slat which is both correct and which has an initial binding strength less than $\frac{G_{mc}}{G_{se}}$.
- Accuracy - calculation of growth accuracy as a decimal. Calculated as $\frac{correctSlats}{totalSlats - seedSlats}$.

- Growth Rate - calculation of growth rate as a float. Calculated as $\frac{correctSlats}{time}$.
- Date - the date this output file was processed for record-keeping purposes.
- Termination - the reason for a simulations termination. If a simulation did not terminate, this is set to 'CONTINUES'. If it terminated, it will be set to either, 'MAX ATTACHMENTS REACHED,' or to, 'MAX STEPS REACHED.'

3 Results and Analysis

3.1 Data and Introduction

With the software architecture in place, this section describes the iterations of simulations run, their results, and the analysis leading to each subsequent set of simulations. While the experimental data set used as a basis of comparison utilized 32-length slats (cooperativity 16 or "v16") [2], other cooperativities were tested as well. Additionally, while each slat in a given macrotile was essentially unique in the physical slat experiments [2], our simulations covered a wider variety of slat counts. Aside from variation of the target parameters G_{mc}, G_{se}, and k_f, sets also varied greatly in the number of simulation steps and values of slat count and cooperativity. The cooperativity and slat count for any experiment run are shown in Table 3.1.

Set Name	Description	Purpose
v4_All	v4 ribbon sets with slat count 4	Kf estimation and framework testing
ribbon-rect-v6-d6	v6 ribbon sets with slat count 6	Kf estimation and framework testing
ribbon-rect-v8-d8	v8 ribbon sets with slat count 8	Kf estimation and framework testing
ribbon-rect-v16-d16	v16 ribbon sets with slat count 16	Kf estimation, G_{mc} and G_{se} narrowing
v16_d2	v16 ribbon sets with slat count 2 and broad range	slat count testing
v16_d2_50-70Gmc	v16 ribbon sets with slat count 2 and Gmc 50-70	Gmc and Gse narrowing
v16_fine	v16 ribbon sets with slat count 16 and finer grain Gmc and Gse	Gmc and Gse narrowing
v16_d2-deep	v16 ribbon sets with slat count 2 and large step depth	slat count testing and ranging
v16_d4-deep	v16 ribbon sets with slat count 4 and large step depth	slat count testing and ranging
v16_d8-deep	v16 ribbon sets with slat count 8 and large step depth	slat count testing and ranging
v16_d16-deep	v16 ribbon sets with slat count 16 and large step depth	slat count testing and ranging
v16_d4_gse9-21	v16 ribbon sets with slat count 4 and larger Gse range	slat count testing and ranging
v16_d8_gse9-21	v16 ribbon sets with slat count 8 and larger Gse range	slat count testing and ranging

Table 3.1: List of Simulation Sets

3.2 Lower Cooperativity Simulations

The first simulation batches completed were at cooperativities below v16, particularly v4, v6, and v8. This was largely due to lower cooperativity simulations resulting in much quicker computation time, giving quicker results for testing the simulator and software framework meaning results pertaining to testing the simulator and framework would come sooner. It also allowed early evaluation and estimation of parameters based on general rules rather than specific values, as it allowed confirmation of some important assumptions about the general relationship between the parameters such as the significance of the proportion between G_{mc} and G_{se}.

Due to the quicker computation of lower cooperativity systems, a v4 set slat system was tested first to get the most comprehensive parameter mapping. G_{mc} values from [0 - 200] and G_{se} values from [0 - 200] were tested in increments of 10, while k_f values from [500 - 90500] were tested in increments of 10,000. The upper boundary for G_{mc} and G_{se} were selected primarily from realistic ceilings explored by previous works with kinetic models [1] [5]. Because k_f was much more nebulous, it had to be tested more thoroughly over a parameter range with no fixed upper bound. The value of α calculated earlier gives us a clue to k_f's possible values, but a broad range was tested regardless. **Figure 3.1** shows the results of the first run of the pipeline. Visualization software created by Dr. Matthew Patitz built on top of the *plotly* Python library plots simulation results along the three axes of the specified parameters. The dependent variables (accuracy and growth rate) are visualized as the color and size of data points. Green dots represent simulations which meet a threshold accuracy, which is defined as the fraction of assembled slats which match the placement of slats in the golden reference system; the threshold value used here is 90%. Red dots represent simulations that did not meet this accuracy threshold. Larger dots represent higher growth rates, while smaller dots represent lower growth rates.

These initial results, besides proving the effectiveness of the pipeline, appear

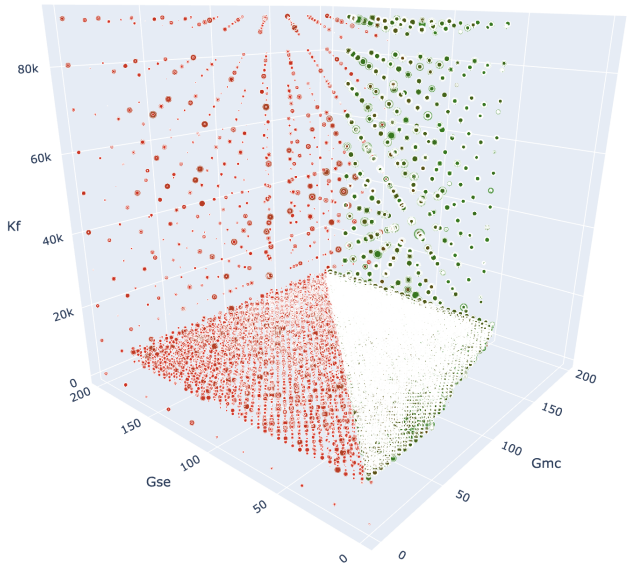

Figure 3.1: Initial test of v4 simulations and software

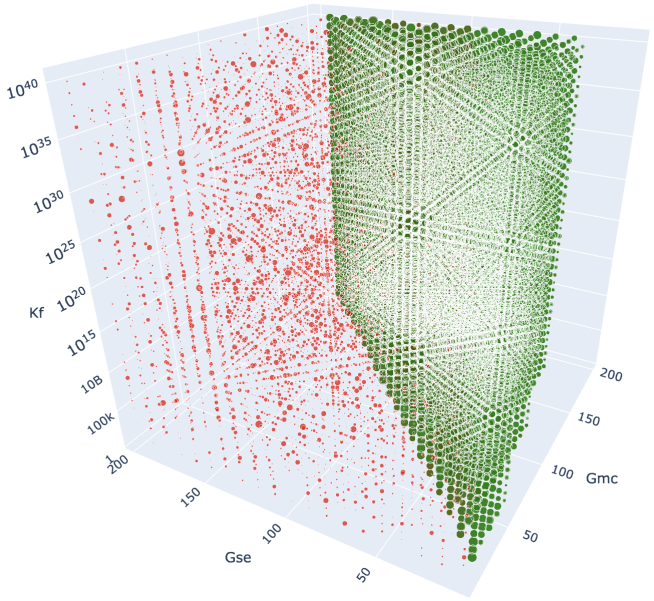

Figure 3.2: Comprehensive test of v6 system testing over an exponential range of k_f

to confirm that 1) k_f is independent of G_{mc} and G_{se}, 2) there are identifiable regions of accurate growth, and 3) a kTAM-analogous relationship between G_{mc} and G_{se} appears to hold for slats as well as tiles.

The next system tested was a v6 system over a larger data set of simulations. These were still quick enough to be run locally over several hours and did not have a significant need for additional computing resources. **Figure 3.2** shows the results of this simulation, which match the pattern of v4 results.

3.3 Calculating k_f to Simplify Parameter Ranges

The next set of experiments leaped to cooperativity 16. At a glance, these simulations appear to hold to the same pattern of a single area of tangible, accurate growth surrounded by areas of tumorous growth or no growth at all. However, the slope of the ratio between G_{mc} and G_{se} appears to be different in **Figure 3.4**. The slopes of each cooperativity generally matched the assumption that the ratio of G_{mc} to G_{se} corresponds to the cooperativity of a system, which is itself representative of the melting temperature beyond which growth rate decreases dramatically.

Once again though, the value of k_f does not appear to affect the overall relationship between G_{mc} and G_{se}. To determine a realistic value for k_f moving forward, it is necessary to revisit the kTAM equations and discuss their applicability to kinetic simulations of slat systems. Recall that what we are referring to as k_f is actually $\hat{k_f}$ in the following equation:

$$\hat{k_f} = u_0 k_f e^{\alpha}$$

Because u_0 and k_f on the right side of the equation are predetermined constants, only α is necessary to calculate our effective rate constant. α appears in another equation stated before:

$$G_{mc} = \alpha - \ln(\frac{[C]}{u_0})$$

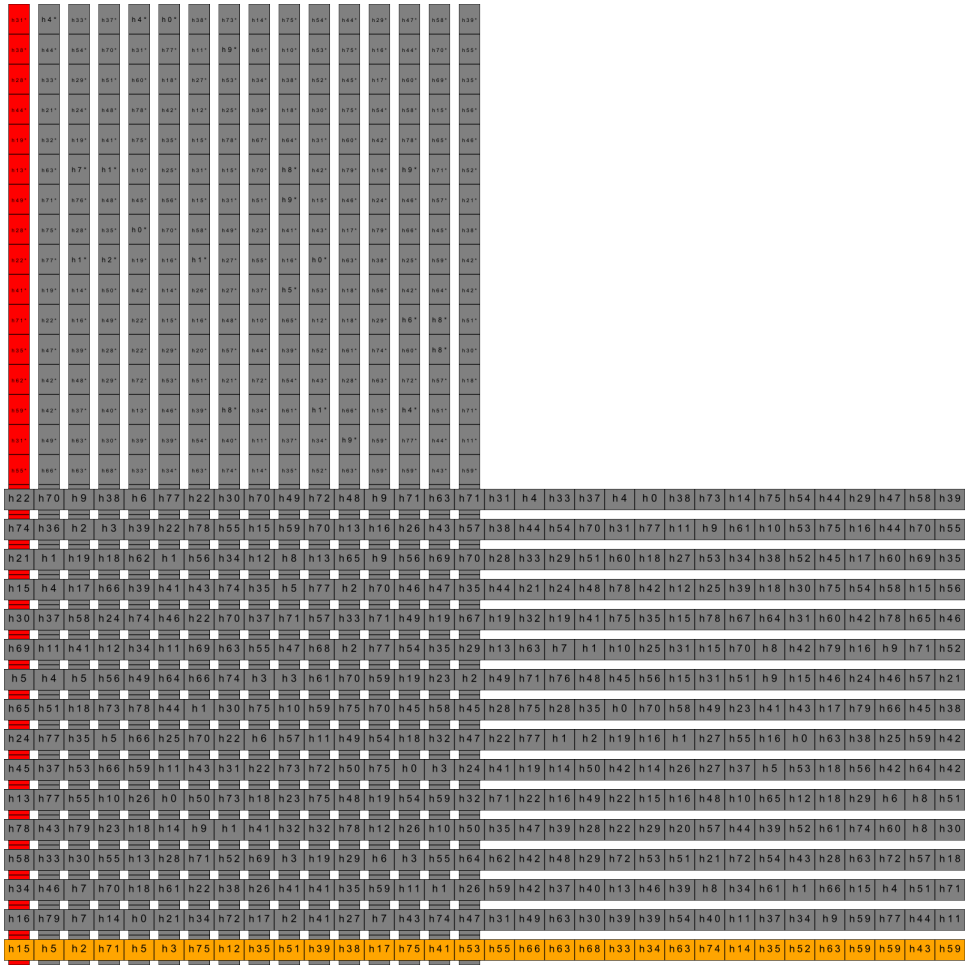

Figure 3.3: Cooperativity-16 system to be tested.

This equation can be solved for α while also substituting in the definition of G_{se} after relating G_{mc} to G_{se} by a factor of cooperativity, corresponding to the melting point of the ribbon slats, following from the ratio discovered by the coarse parameter mapping:

$$G_{mc} = 16(G_{se}) \qquad\qquad G_{se} = \frac{-\Delta H^{\circ} + T\Delta S^{\circ}}{RT}$$

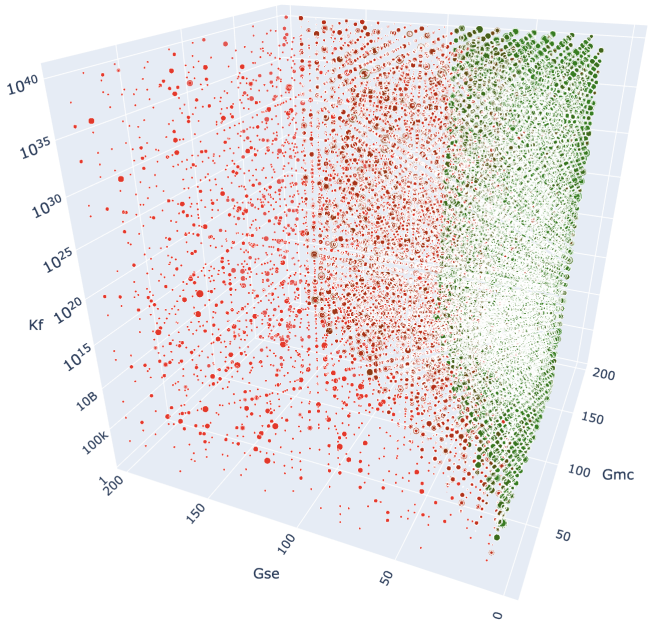

Figure 3.4: First test of cooperativity-16 systems to determine if the relationship between variables holds when cooperativity is scaled up significantly.

$$\alpha = ln(\frac{[c]}{u_0}) + 16(\frac{-\Delta H^\circ + T\Delta S^\circ}{RT})$$

Remember that $[c]$ is the concentration of tiles (here slats), and it has been set to 20 nM in past lab experiments. u_0 is a constant used for unit conversion and is equal to 1 M. This means that the first term of our alpha equation is a constant. In the kTAM, the values of ΔH° and ΔS° are calculated using the nearest neighbor model, and we use $-45\frac{kcal}{mol}$ and $-0.135\frac{kcal}{mol(K)}$ as typical values for nucleotide handles of length 7 based on Dr. Shih's slat ribbon data [2]. The optimal growth temperature (T_{opt}) and melting temperature (T_{melt}) were calculated to be $30^\circ C$ and $42^\circ C$ respectively. As stated in a previous section, the main hypothesized difference between the kTAM and slat kinetics is the value α. Now that all values of the right side of that equation are known, we find that $\alpha \approx 40$. It should be noted that this method of calculating α is very rough, and is only meant to produce a

reasonable value of α that can be used to simulate results that are reasonably close to Dr. Shih's experimental data.

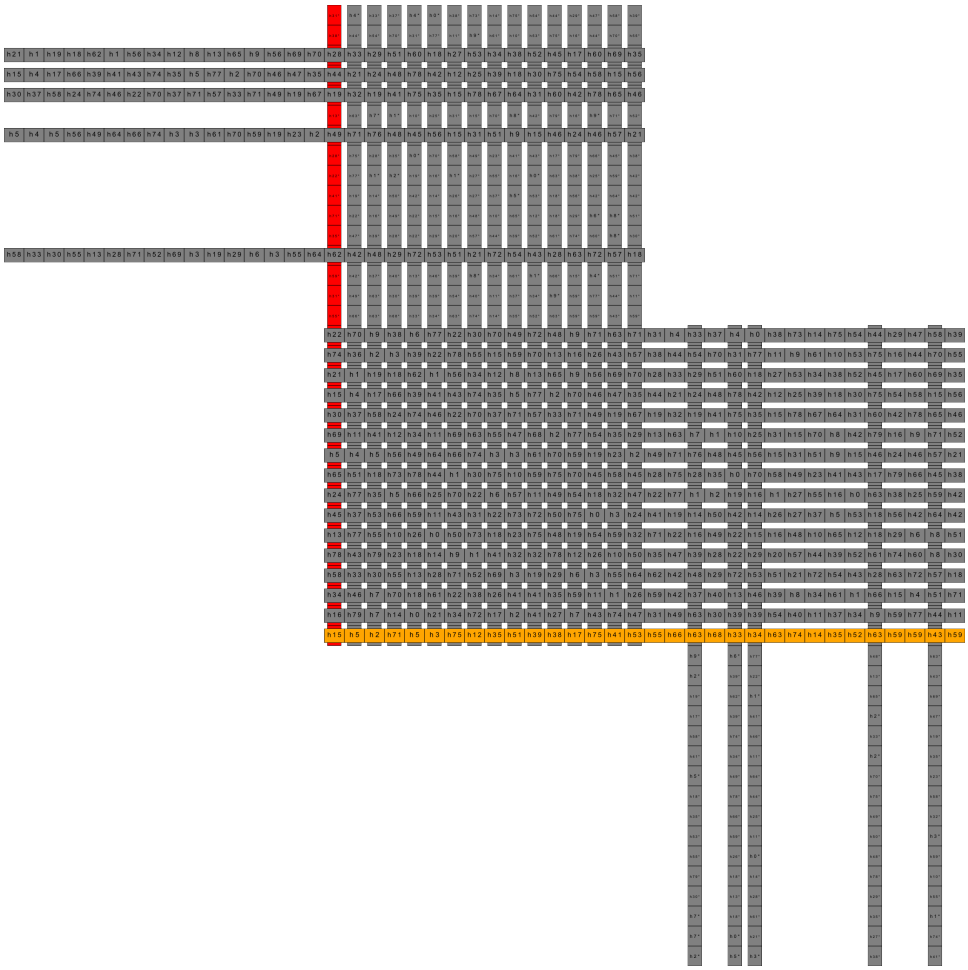

Figure 3.5: Example ribbon growth for a system of v16 slats.

3.4 Higher Cooperativity Simulations

For the rest of the experiments performed, k_f will assume a fixed value of $2.4 * 10^{23}$. The simulation speedup accompanying this decision is significant! Now, combinations of parameters from only two variable ranges need to be selected rather than from three, reducing the number of possible combinations to be tested

by an order of magnitude.

With k_f fixed, the next set of v16 simulations proceeds over the following parameter range: $G_{mc} \in [50, 70]$, $G_{se} \in [1, 9]$. Because of the computational time savings that accompany a fixed value of k_f, the other two parameters may be varied at a finer granularity to generate a more extensive map of assembly characteristics.

Given a fixed value of k_f, it is much quicker to run simulations for very fine-grained ranges of G_{mc} and G_{se}, so the focus of simulations shifted towards creating thorough maps of the ranges for these parameters that produced accurate growth.

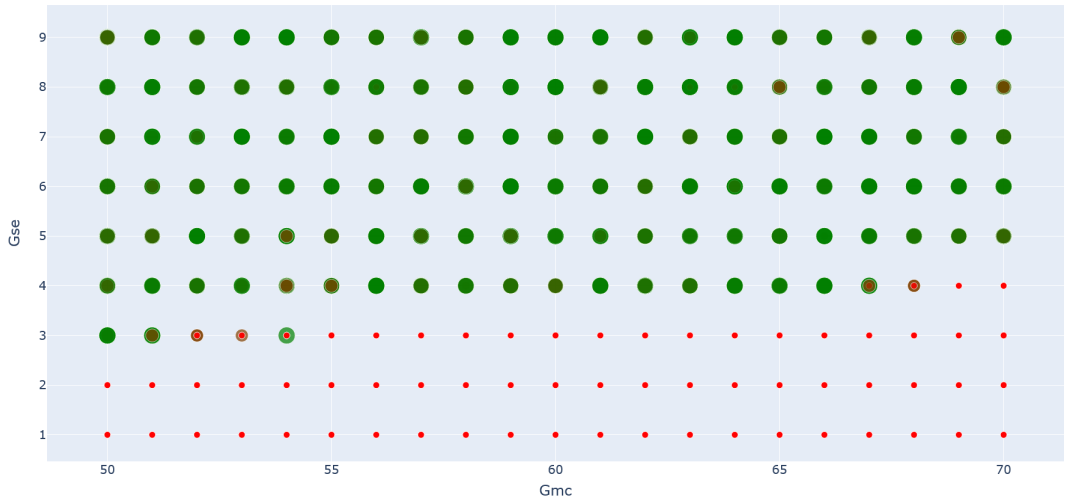

Figure 3.6: First large-scale simulations with a fixed value of k_f. The computation time saved allows for finer-grained testing of the remaining parameters.

3.5 Slat Count Experiments

Following the determination of an appropriate value for k_f and the identification of good ranges for G_{mc} and G_{se}, the focus of simulations shifted to comparisons of the range of G_{se} at varying slat counts within the same cooperativity at a fixed K_f and G_{mc}. The previous ranges for G_{mc} and G_{se} were calculated under the general assumption of slat count 16, meaning that the only realistically possi-

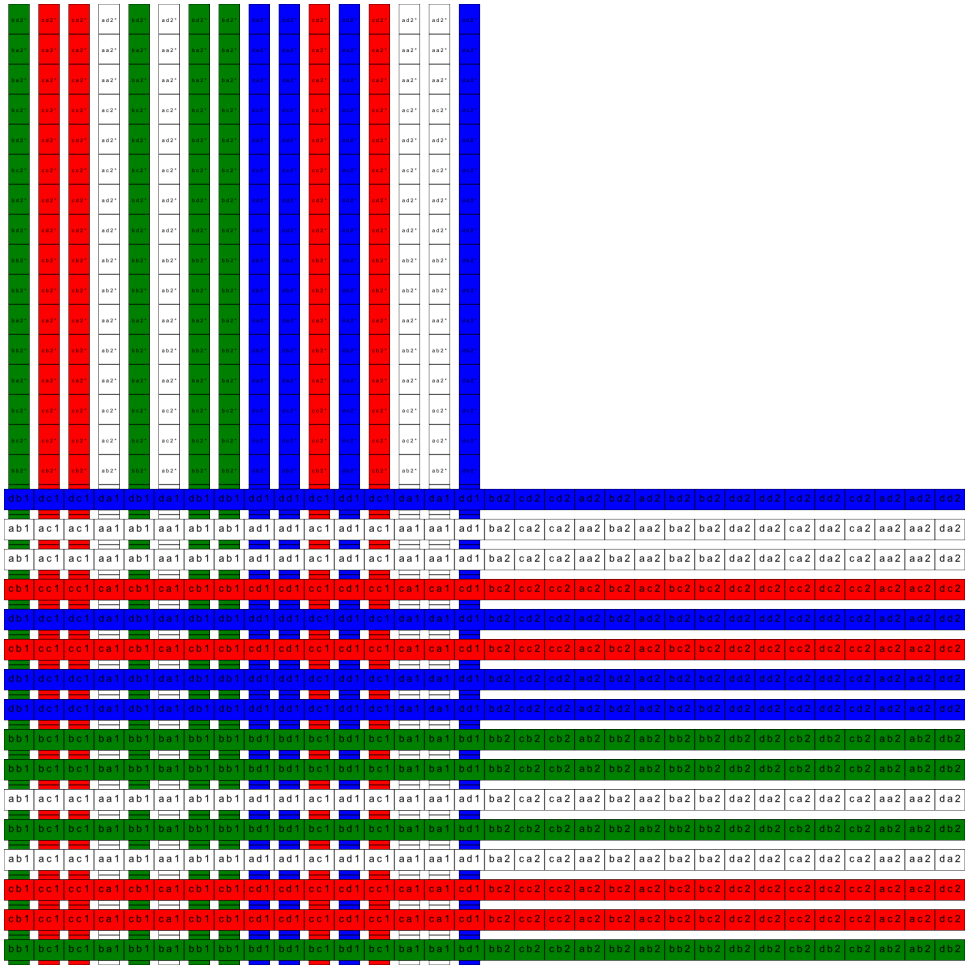

Figure 3.7: System with only 4 unique slats in each direction.

ble errors in slat attachment have a binding strength of only 1 as long as enough unique glue handles are used. This generally means that higher slat counts are more accurate as the range of possible errors they can produce are guaranteed to be weaker than those possible in a lower slat count, and thus are more likely to detach from the superstructure. This can result in a wider range of values of G_{mc} and G_{se} which produce accurate growth.

However, there are a few advantages to be found in lower slat counts as well. It is generally expected intuitively that lower slat counts would assemble

more quickly due to fewer unique types of slats. Because the forward rate is the same for correct and incorrect attachments alike, lowering the number of slats that must attach and detach before finding a correct match is hypothesized to increase the growth rate while having negligible affect on error rate. The reduction in the number of unique slats also reduces the monetary cost of physical experimentation, as well as the computational cost of simulation. Slat counts were tested in, at first, identical ranges of G_{mc} and G_{se} and the same set value of k_f at slat counts of 2, 4, and 8.

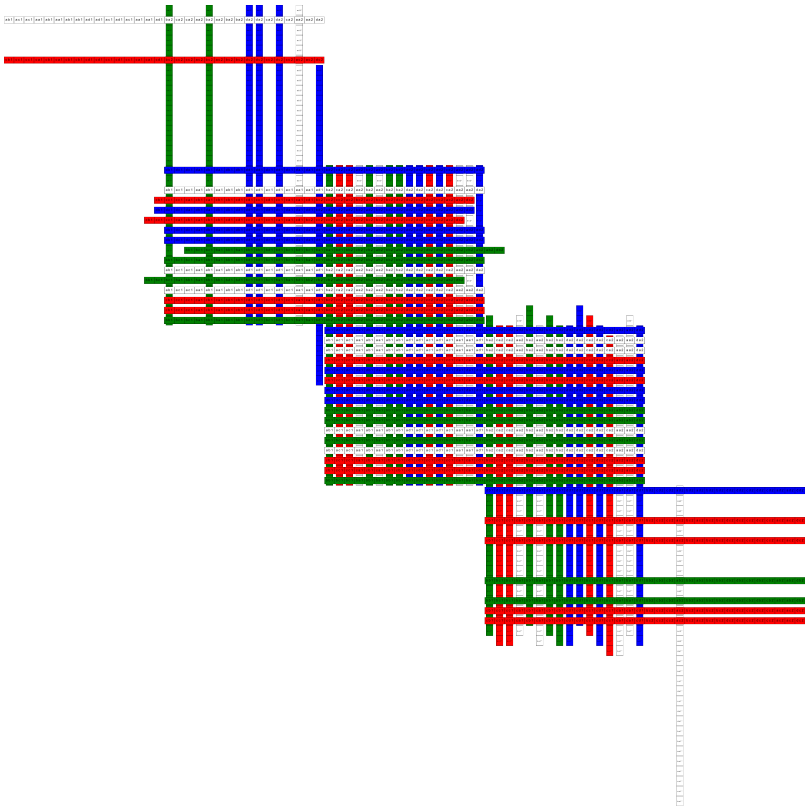

Figure 3.8: Example of v16 system with fewer unique slats post-simulation.

Therefore, there is strong reason to want to study the differences between slat counts of the same cooperativity. Although the differences in accuracy and growth rate are generally expected to be marginal, a much wider variance in the good ranges of G_{mc} and G_{se} was expected.

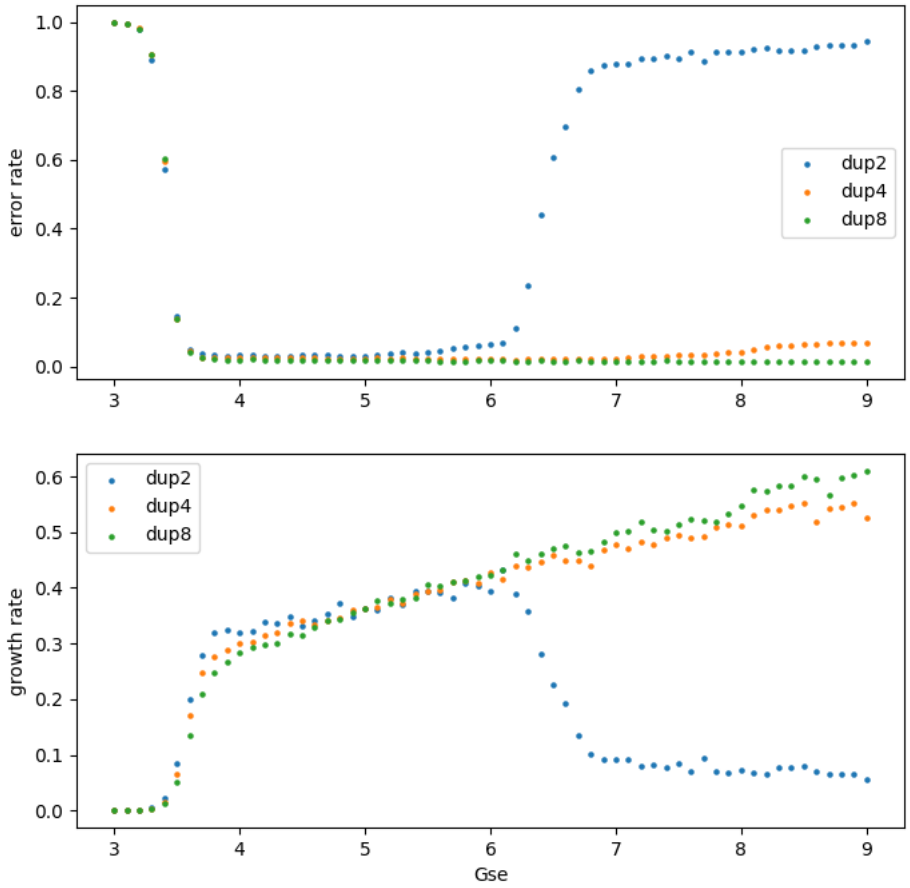

Figure 3.9: Simulation results for various v16 slat counts over a fixed range of parameters.

As seen in Figure 3.9, many of the general assumptions about the effect of slat count seem to hold. While the impact on accuracy is marginal at best, the most important thing to glean from the figures is the apparent narrowing of the good range of G_{se} observed in slat count 2. It was also thought that a lower slat count would increase the growth rate since any given tile, comprising a larger fraction of the slat types, would be more likely to find its correct match. Initially,

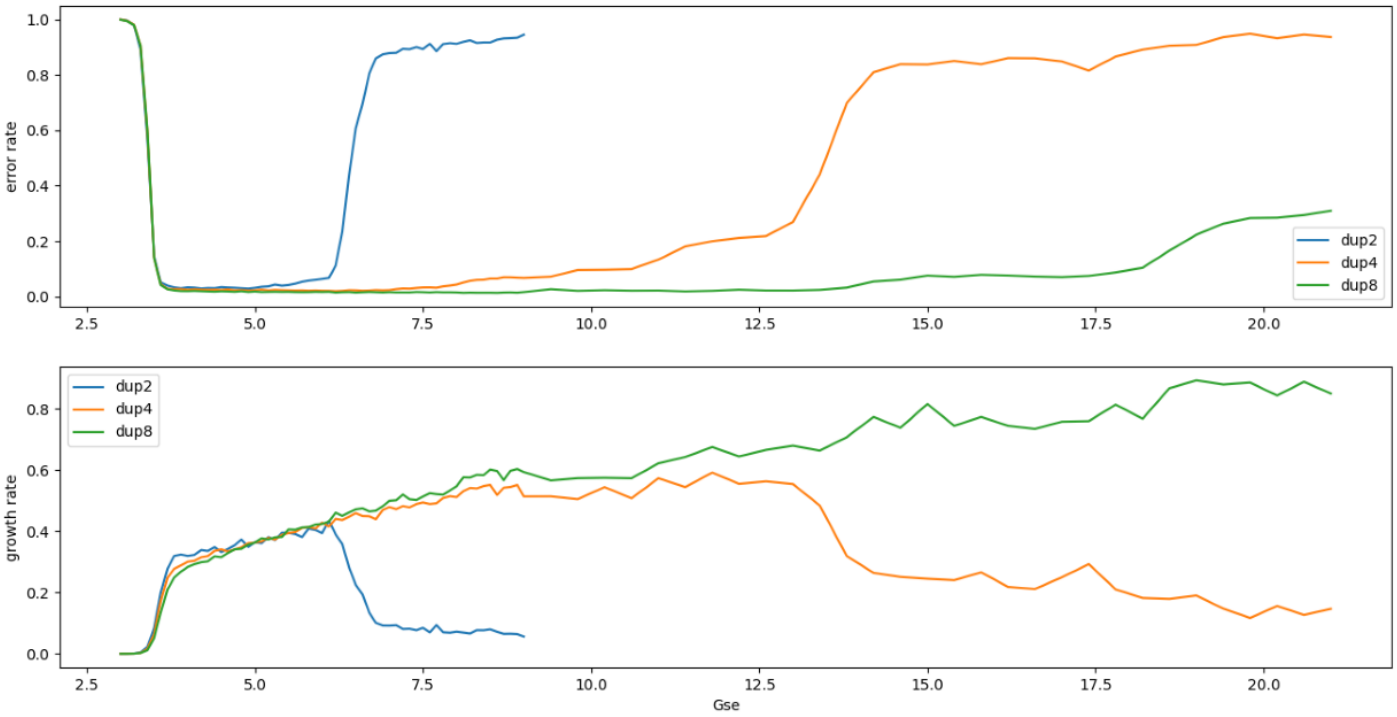

Figure 3.10: Extended results for various v16 slat counts when the error-resilient zone of accuracy was larger than expected.

this does not appear to occur to a significant extent. As shown in Figure 3.10, simulations were then run with wider ranges for G_{mc} and G_{se}, with this narrowing now becoming apparent also at slat count 4. The "dupX" in these figures is an older naming convention for the slat count discussed today.

4 Conclusion

This project determined through simulation and comparison to experimental data appropriate ranges for the values of G_{mc}, G_{se}, and k_f primarily at a cooperativity of v16 using polyomino slats. Additionally, methodologies for repeating this procedure for different cooperativities were established. The ultimate conclusion of these simulations is that many of the assumptions and relationships established by previous definitions of the kinetic model generally hold for polyomino slats, with deviations caused by variation in the value of α and, by extension, k_f. After a proper value of α and k_f were calculated, suitable ranges of G_{mc} and G_{se} that produce accurate, controlled growth were found in ratios similar to those predicted by the kTAM.

Further work would consist of more extensive testing of other cooperativities as well as testing of different assemblies aside from Shih's ribbons. Additionally, an assessment of the simulator's predictive power by recreating physical experiments with conditions corresponding to simulator parameters could be explored.

Bibliography

[1] E. Winfree, "Algorithmic self-assembly of dna," Ph.D. dissertation, California Institute of Technology, 1998.

[2] D. Minev, C. M. Watersinger, A. Ershova, and W. M. Shih, "Robust nucleation control via crisscross polymerization of highly coordinated slats," *Nat Commun 12, 1741*, vol. 12, 2021.

[3] M. J. Patitz, "An introduction to tile-based self-assembly and a survey of recent results," *Natural Computing*, 2014.

[4] P. W. K. Rothemund, E. Winfree, and N. Papadakis, "Algorithmic self-assembly of dna sierpinski triangles," *PBIO*, 2004.

[5] C. Evans and E. Winfree, "Physical principles for dna tile self-assembly," *Chem. Soc. Rev.*, vol. 46, pp. 3808–3829, 2017. [Online]. Available: http://dx.doi.org/10.1039/C6CS00745G

[6] E. Winfree, "The xgrow simulator," last accessed 27 April 2023. [Online]. Available: https://www.dna.caltech.edu/Xgrow/

[7] M. J. Patitz, "Simulation of self-assembly in the abstract tile assembly model with isu tas," 01 2011.