University of Arkansas – CSCE Department
Capstone I – Final Proposal – Fall 2021

Designing and Simulating the Self-Assembly of DNA Nanostructures and DNA Computers

Lindsey Albin, Ben Hughes, Kyle Sadler, Christopher Souvanouphong

Abstract

The design of DNA strands is an expensive and time-consuming process. Researchers in the nanotechnology field waste valuable resources transforming nanostructure designs into DNA strands. The goal of the project is to simplify the process of constructing DNA strand designs by creating a quick and easy way for researchers to convert nanostructure tile designs into DNA strand diagrams.

Approximately a dozen labs around the world have the capability of implementing tile-based self-assembly. The successful implementation of this project can impact these labs by reducing consumable and labor costs, allowing rapid development of new designs, and allowing the quick export of a design for simulation to test strains and their stability.

1.0 Problem

DNA tile-based self-assembly is a process in which a collection of simple yet disorganized components coalesce to form complex structures. While this happens naturally all around us (take a snowflake for example), scientists have developed methods to artificially manipulate matter on the atomic level to replicate this phenomenon. For example, the abstract Tile Assembly Model (aTAM) is a high-level abstraction that ignores potential errors. The aTAM is a mathematical model used to implement tile sets via designed DNA strands. In this model, assembly starts from a given “seed” tile and grows non-deterministically and asynchronously.

One of the biggest issues facing researchers today when working with tile-based self-assembly is the cost overhead and the extensive number of software needed for these models. The current process of transforming a nanostructure design into a set of DNA strands is tedious and requires many independent software packages. This means that researchers must waste valuable time importing, exporting, transferring, and reformatting data. The other problem faced is the cost of self-assembly. Ordering the DNA in a tile-based format can get expensive very fast. If there was a way for scientists to minimize the number of errors and mismatches in the DNA strands, it could save them thousands of dollars in the long run.

2.0 Objective

The objective of this project is to simplify the DNA nanostructure design process. Instead of having to use multiple independent software packages, our project will allow researchers to easily transform an abstract nanostructure tile assembly design into a DNA strand diagram, readable by Scadnano, with the click of a button. Scadnano is web-based software for visualizing DNA molecules as strand diagrams. Our software will be hosted on a web server so that it is accessible and does not require the user to manually install and compile software packages.

3.0 Background

3.1 Key Concepts

DNA Base Pairs. DNA base pairs are molecules called nucleotides, on opposite strands of the DNA double helix. They form chemical bonds with one another and act like rungs in a ladder to hold the two strands together. The four bases are adenine (A), cytosine (C), guanine (G), and thymine (T). Adenine forms a pair with thymine, and guanine forms a pair with cytosine. The binding of these pairs forms the structure of DNA.

DNA Nanotechnology. DNA nanotechnology is the study and design of DNA nanostructures. Strands of synthetic DNA can be designed so that they bind in a controlled and predictable process which allows arbitrary DNA structures to be created at the nanoscale. This is possible due to two technological advancements: a detailed understanding of the DNA binding process and the ability to manufacture synthetic DNA. Nanoscale DNA structures have applications in fields such as biophysics, diagnostics, nanoparticle and protein assembly, biomolecule structure determination, drug delivery, and synthetic biology [3].

DNA Tile Assembly. This project focuses on a DNA nanostructure fabrication process called *DNA tile assembly*. In tile assembly, DNA strands are conceptually organized into rectangular structures called *tiles* which are the fundamental building unit in tile assembly structures [Figure 1]. These tiles have four DNA binding domains (a, b, c, d) which are used to control the binding properties of the tile and determine the shape of the fabricated nanostructure. Multiple types of tiles can be designed so that when mixed in a solution, tiles attach to each other in a way that builds the intended DNA nanostructure. The entire set of tile types used in an assembly is called a tileset.

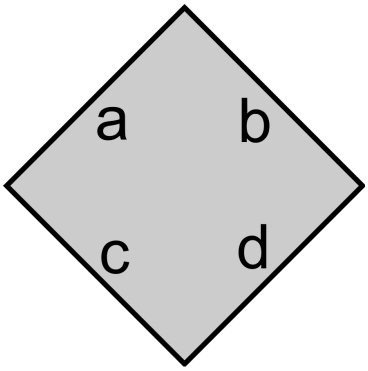

Figure 1. Abstract DNA tile used in tile assembly.

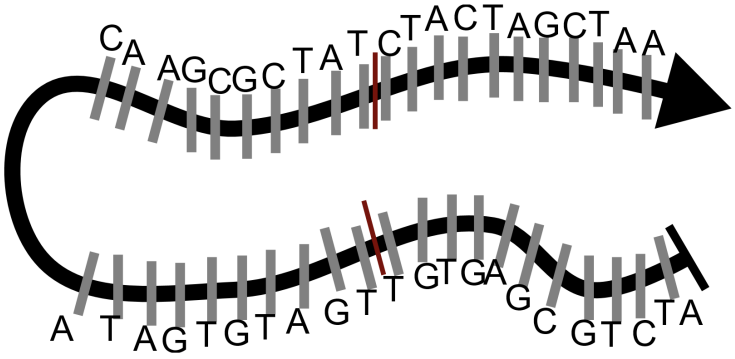

Figure 2. A single-stranded tile formed out of a single strand of DNA.

Figure 3. A core tile which contains two additional DNA strands which brace the structure and form its core.

DNA Tile Motifs. As tiles are merely conceptual representations of blocks of DNA binding domains, there are many ways to implement them in actual strands of DNA. The most common DNA tile motif is *single-stranded tiles*, in which each tile is physically realized as a single strand of DNA bent back on itself [Figure 2]. The other main motif is *core tiles*, in which two additional strands are added to the single-stranded tile to brace the structure and add more stability [Figure 3]. In this project, we handle both single-stranded tile and core tile motifs.

Scadnano. Scadnano is a software package for visualizing DNA molecules as strand diagrams and designing synthetic DNA nanostructures. It allows users to sketch out DNA nanostructures using strand diagrams via a web-based GUI or programmatically through a Python library. Scadnano supports exporting of the nanostructure file for coarse-grained molecular simulation, and simulation of twist and strain on the nanostructure.

3.2 Related Work

There are many existing software applications that are related to DNA nanotechnology and nanostructure design. When designing a tile assembly nanostructure, one of the first steps is to create a tileset using a tile assembly simulator [5]. First, the nanostructure is divided into tiles. The tile assembly simulator then helps simulate how a DNA tileset design will act when the tiles are physically created out of DNA and mixed in a test tube.

Tilesets are often visualized in Scadnano which is a software program for visualizing DNA molecules as strand diagrams [4]. While Scadnano is a useful tool, it is often time-consuming to import tileset designs.

The specific sequences of DNA used in a tile assembly are often picked using a sequence designer such as DNADesign [6]. However, these tools do not make it easy to combine these sequences with a tileset structure and Scadnano, which is the problem our project will solve.

4.0 Design

4.1 Application Requirements

- Input an abstract tile assembly, its name, and its motif style attribute; a strand diagram for motifs that contains information about the bond structure of a single tile including strands involved and how they attach to one another; and a strand CSV file where each strand contains a strand name, tile name, motif type, role in the motif, and the sequence
- All three inputs are required for the software to compute the DNA strand diagram
- Output a DNA strand diagram that can be downloaded by the user and/or opened on the Scadnano website
- If the user chooses to open the computed DNA strand diagram in Scadnano, the application must link to the Scadnano website and open the DNA strand diagram for the user
- Software must be hosted on a web server
- A nice and clear transition from our webpage to Scadnano

- The user interface must be easy to understand and easy to use
- The process of inputting all of the data must be clear, organized, and provide the user with feedback on correct and incorrect inputs
- The user interface must keep the user's attention (show a loading bar, etc) while the computation is taking place

4.2 High-Level Architecture

The architecture will consist of two primary components: the React user interface and the Python backend.

User Interface

The frontend will be built using and React and will be responsible for providing the user interface to upload the three input files: the abstract tile assembly, the tile strand diagram, and the strand configuration CSV [Figures 4, 6]. It will be responsible for navigating the user through the process, for providing feedback on incorrect uploads or errors, and for helping the user download and/or open their computed DNA strand diagram. It will also be responsible for calling our backend API to process the inputs and display the inputs and outputs of our program [Figure 5].

Python Backend

The main logic of our program will be written in Python. This design choice was made because the Scadnano API is only accessible through a Python package. When the user uploads the input files, the backend will be responsible for receiving the files, processing them into the resulting strand diagram, and sending the processed file back to the user.

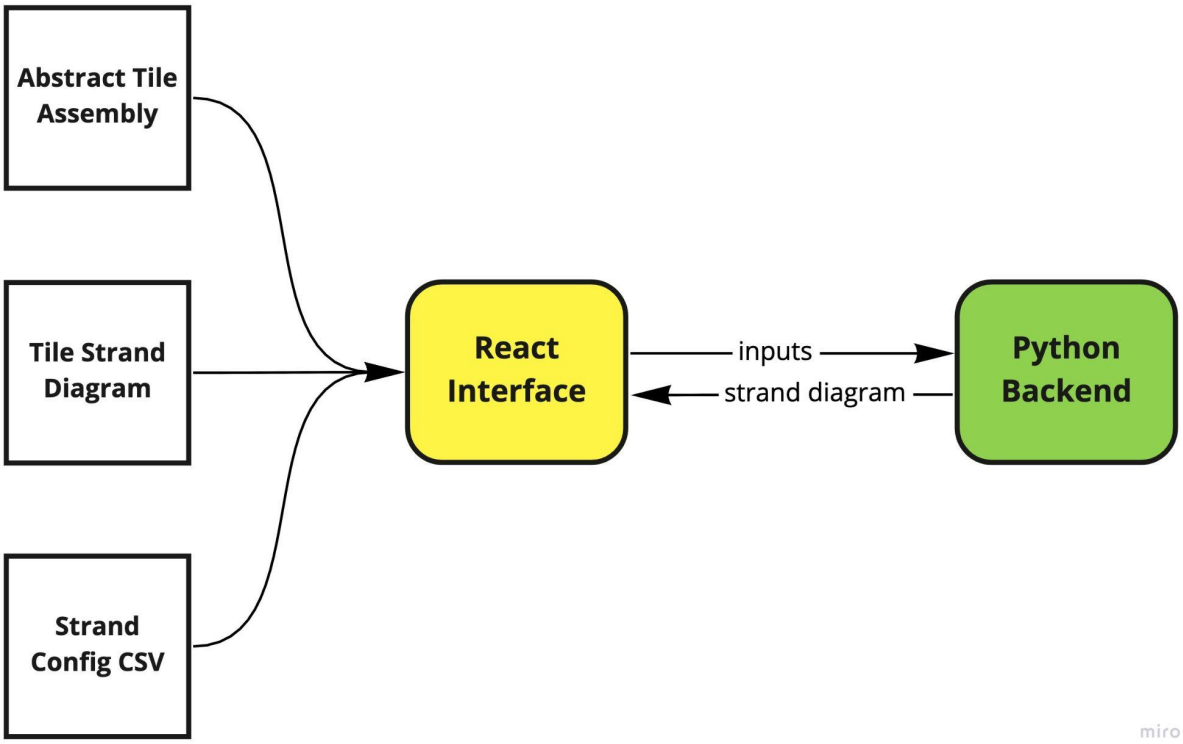

Figure 4. Proposed application architecture.

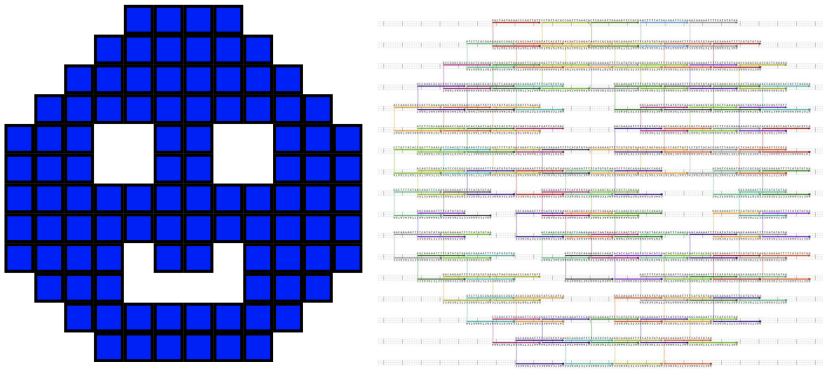

Figure 5. Example abstract tile assembly input (left) and resulting output strand diagram (right).

```
# Name,TileSequence,Yeild,Purification
Tile0101,CCAGTACAGT ATAGAAACCGAATGTTCCGGCA GCTGCTTTCT CTTCTTGTCG ACCCGATATTGGTCTCCCTGTT GCAGTAGTCG,100nM,STD
Tile0102,CGACAAGAAG TAGCCGAATTGACGCTACGAA CGACTACTGC TCTGAACTGT AGATAGGATTGTGCCCATCGA GCTCTCATGG,100nM,STD
Tile0103,ACAGTTCAGA ACAAACATGCAGCGTATTCGGA CCATGAGAGC CGAACAATCT TACCGTAGTGCAGGTCCAGTTT GCAGACAAGT,100nM,STD
Tile0104,AGATTGTTCG AAACTTCACAAGGAGCGTGAA ACTTGTCTGC ACTTCATTCG TTAAGTGGTGTCACGCTGGAA CAGACAGAGC,100nM,STD
Tile0105,CGAATGAAGT TCCAATACGGGATCCCGAACAA GCTCTGTCTG ACTCTGTTCG AATTTAGAGCGCCGTGCATCTT GCTCAGACTC,100nM,STD
Tile0106,CGAACAGAGT TTCGTAGAAACGGCTGCCTAA GAGTCTGAGC GTGTTAGTCG AACTTGTGCTGTCGAGAGAGA GGAAGACAGT,100nM,STD
Tile0107,CGACTAACAC AGTTTCAGCAGCCGAACGATTA ACTGTCTTCC CGATATCTGC AATATCCGGCTTCCCGCTAAGA ACAGATCAGT,100nM,STD
Tile0108,GCAGATATCG ATGGTTAAGCCTTGCGTTTGT ACTGATCTGT GCTGCTAGTC TTTGAGTGGGTGGCTTCTCTA CTCAGACAGA,100nM,STD
Tile0109,GACTAGCAGC TTTCCGGACAATCTTTGCGACA TCTGTCTGAG GCTGATGTCG ATTCCTGACCTCACCTGCCTTA ACAGTACAGC,100nM,STD
Tile0110,CGACATCAGC TCTCCTTAGACAACACGGGTT GCTGTACTGT GCTCTCTTGT TACTCGACGTGATTTGCCTGA TTCATCAGCA,100nM,STD
Tile0111,ACAAGAGAGC ATCCGCTATGTGTCGGCAGTAT TGCTGATGAA GTGATGTTCG TAGAGCCCTGAAACCCTCAAGT CCAGATCAGC,100nM,STD
Tile0112,CGAACATCAC AGTCTGTCCGGGCAAATCTTT GCTGATCTGG ACTGTACTGG TGTTGACGAAGACTCCGTGTT AGAAAGCAGC,100nM,STD
Tile0101coreTop,TGCCGGAACATTCGGTTTCTAT,25nM,STD
Tile0102coreTop,TTCGTAGCGTCAATTCGGCTA,25nM,STD
Tile0103coreTop,TTCGAATACGCTGCATGTTTGT,25nM,STD
Tile0104coreTop,TTCACGCTCCTTGTGAAGTTT,25nM,STD
Tile0105coreTop,TTGTTCGGGATCCCGTATTGGA,25nM,STD
Tile0106coreTop,TTAGGCAGCCGTTTCTACGAA,25nM,STD
Tile0107coreTop,TAATCGTTCGGCTGCTGAAACT,25nM,STD
Tile0108coreTop,ACAAACGCAAGGCTTAACCAT,25nM,STD
Tile0109coreTop,TGTCGCAAAGATTGTCCGGAAA,25nM,STD
Tile0110coreTop,AACCCGTGTTGTCTAAGGAGA,25nM,STD
Tile0111coreTop,ATACTGCCGACACATAGCGGAT,25nM,STD
Tile0112coreTop,AAAGATTTGCCCGGACAGACT,25nM,STD
Tile0101coreBot,AACAGGGAGACCAATATCGGGT,25nM,STD
Tile0102coreBot,TCGATGGGCACAATCCTATCT,25nM,STD
Tile0103coreBot,AAACTGGACCTGCACTACGGTA,25nM,STD
Tile0104coreBot,TTCCAGCGTGACACCACTTAA,25nM,STD
Tile0105coreBot,AAGATGCACGGCGCTCTAAATT,25nM,STD
Tile0106coreBot,TCTCTCTCGACAGCACAAGTT,25nM,STD
Tile0107coreBot,TCTTAGCGGGAAGCCGGATATT,25nM,STD
Tile0108coreBot,TAGAGAAGCCACCCACTCAAA,25nM,STD
Tile0109coreBot,TAAGGCAGGTGAGGTCAGGAAT,25nM,STD
Tile0110coreBot,TCAGGCAAATCACGTCGAGTA,25nM,STD
Tile0111coreBot,ACTTGAGGGTTTCAGGGCTCTA,25nM,STD
Tile0112coreBot,AACACGGAGTCTTCGTCAACA,25nM,STD
```

Figure 6. Example strand file.

4.3 Risks

Risk	Risk Reduction
User concerns about research confidentiality	Provide a disclaimer that strand files are never stored on our servers and DNA research intellectual property remains with the user
Research liability for software bugs	Provide a disclaimer that we are not responsible for any incorrect results and all results should be manually checked before using as research material.

4.4 Tasks

1. Gain proper background knowledge required for the project
    a. Understand Scadnano, the software used for visualizing DNA molecules as strand diagrams
    b. Understand the process of creating DNA strands with the Abstract Tile Assembly Model (aTAM)
2. Designing the software
    a. Design the frontend with React
        i. The user interface we intend to create will handle the inputs of the abstract tile assembly, strand diagram, and strand CSV file configuration. The formats of the abstract tile assembly and strand CSV files have already been developed. The abstract assembly file is a simple file format. It starts with a few comments and then for every position in the assembly, the

name of the tile and its position in Z^3 (the 3D integer lattice) are listed. For the project, we will not be dealing with 3D assemblies, so the third coordinate will always be zero. The top of the strand file has the name, tile sequence, yield, purification. Then, below is a list of the tiles [Figure 6]. The important fields for our project are the name and sequence fields. We plan to start with the set format for the project and then modify the format of this file to something more convenient for our program.

ii. The user interface will also display the Scadnano strand diagram by linking to the Scadnano program and opening the file

b. Design the backend with Python
   i. Finalize a list of API routes and their interfaces to
      1. Upload input files
      2. Access status of submitted job
      3. Access the results of the program
   ii. Choose a Python web framework
   iii. Design DNA strand routing algorithm

3. Implementation of the software
   a. For this stage, we will split into two teams: one to implement the backend and one to implement the frontend.
   b. Frontend implementation
      i. Create file uploads for the abstract tile assembly file, strand diagram file, and the strand CSV file
      ii. Add user input validation and helpful error messages if the input is invalid
      iii. Create a loading animation and screen while the computation is taking place
      iv. Create a button that allows the user to download the output Scadnano file
      v. Create a button that allows the user to open the output Scadnano file on the Scadnano website
      vi. Create a button that takes the user back to the start page to upload more files
   c. Backend implementation
      i. Set up a basic Python server that can respond to requests
      ii. Write functions to receive uploaded input files
      iii. Write functions to parse the input abstract tile assembly, strand diagram, and strand CSV files
      iv. Write a function to take the input data and compute the resulting Scadnano strand diagram using the Scadnano Python library
      v. Write a function to write the output Scadnano data to a file
      vi. Create an API route to access the status of a submitted job
      vii. Create an API route to access the results of a submitted job
4. Integration of the frontend with the backend API
   a. Ensure that files and input data can be uploaded from the frontend to the backend
   b. Ensure that output data from the backend can be sent to the frontend both as JSON and as a file
5. Testing the software

a. In this stage, we will test the software's features with different inputs and make design changes where needed

4.5 Schedule

Tasks	Dates
Background research on the Scadnano libraries, software, etc	1/10 - 1/14
Make mockups of frontend pages in Figma	1/17 - 1/21
Setup Python server and write build scripts	1/24 - 1/28
Implement front-end file upload and input validation	1/31 - 2/11
Define API routes, interfaces, and behavior	2/14 - 2/18
Write backend functions to parse input files and data	2/21 - 2/25
Design DNA strand routing algorithm	2/28 - 3/11
Implement backend API routes and functions	3/14 - 3/25
Integrate front-end interface and backend API	3/28 - 4/8
Polish user interface and add ease-of-use improvements	4/11 - 4/15
Clean up code and work on the final presentation	4/18 - 4/22
Presentation and final report	4/25 - 4/29

4.6 Deliverables

1. Design Documentation: a document outlining the architecture and tech stack of this project and how each component works.
2. Python Backend: Python script that transforms abstract tile assemblies into DNA strand models.
3. React Frontend: The Javascript files that make up our web interface for this project and allow for file uploads
4. Build Scripts: scripts to build and launch the application on a web server
5. Live Web Address: URL to live version of our project
6. Final Report

5.0 Key Personnel

Kyle Sadler – Sadler is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. He is also pursuing a Bachelor of Science in pure mathematics from the University of Arkansas. He currently works full-time for Pipedream, a robotics startup building underground delivery robots. He will be responsible for the application architecture and backend.

Lindsey Albin – Lindsey is a senior Computer Science and Graphic Design major with minors in Mathematics and Art history in the Computer Science and Computer Engineering Department and the Graphic Design Program at the University of Arkansas. She currently works at the McMillon Innovation Studio on campus as the student creative director and previously she worked teaching kids to code using Kotlin and Minecraft. She has taken a class on user experience design and computer graphics along with programming paradigms where she learned Python. She will be responsible for the front end.

Ben Hughes – Hughes is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has previously had an internship with the Walt Disney Company. He is currently working on the front end / UI for a mobile application in Mobile Programming. He has worked on Javascript and Python in the Paradigms course. He will be responsible for the front end.

Christopher Souvanouphong – Christopher is a senior Computer Science major at the University of Arkansas and pursuing a minor in Mathematics. He has previously developed a mobile ordering application for a local restaurant, allowing him to gain experience with both frontend and backend development. He will be responsible for the backend development.

Dr. Trent Rogers – Dr. Rogers is a postdoctoral researcher at Maynooth University researching self-assembling and self-organizing systems. He was a National Science Foundation Graduate Research Fellow and the recipient of the Doctoral Academy Fellowship as a Ph.D. student. He graduated from the University of Arkansas with a Ph.D. in computer science in 2019.

6.0 Facilities and Equipment

Heroku Account - A Heroku account will be used to host our application.

7.0 References

[1] DNA Nanotechnology, https://www.nature.com/articles/natrevmats201768

[2] Scadnano, https://scadnano.org/

[3] Tile Assembly Simulator, http://self-assembly.net/mpatitz/papers/TAS-SASOW.pdf

[4] DNADesign, https://www.dna.caltech.edu/DNAdesign/

[5] Definition of base pair, https://www.cancer.gov/publications/dictionaries/cancer-terms/def/base-pair