Abstract

Self-assembly is a term used to describe the process of a collection of components combining to form an organized structure without external direction. The unique properties of double-stranded DNA molecules make DNA a valuable structural material with which to form nanostructures, and the field of DNA nanotechnology is largely based on this premise. By modeling nanostructures with discrete graphs, efficient DNA self-assembly becomes a mathematical puzzle. These nanostructures have wide-ranging applications, such as containers for the transport and release of nano-cargos, templates for the controlled growth of nano-objects, and in drug-delivery methods. This research project centers around the exploration of the graph theoretical and combinatorial properties of DNA self-assembly, as well as development of computational tools to aid in answering fundamental questions that arise.
1 Project Description

1.1 Introduction and Background

Motivated by the recent advancements in nanotechnology and the discovery of new laboratory techniques using the Watson-Crick complementary properties of DNA strands, graph theory has become useful in the study of self-assembling DNA complexes. DNA self-assembly, and self-assembly in general, is a rapidly advancing field, with [24, 27] providing good overviews. Synthetic DNA molecules have been designed that self assemble into given nanostructures, starting with branched DNA molecules [14, 31], nanoscale arrays [32, 33], numerous polyhedra [3, 9, 10, 28, 39], arbitrary graphs [13, 25, 34], a variety of DNA and RNA knots [20, 21, 30], and the first macroscopic self-assembled 3D DNA crystals [40]. This has led to molecular scaffolding made of DNA which have wide-ranging potential, such as containers for the transport and release of nano-cargos, templates for the controlled growth of nano-objects, biomolecular computing, biosensors and in drug-delivery methods (see [1, 6, 7, 15, 18, 22, 23, 38, 26, 36]).

Furthermore, new experiments demonstrating the feasibility of engineering synthetic DNA nanoscale constructs offer great promise for emergent applications in nanoelectronics, biosensors, biomolecular computing, drug delivery systems, and directed organic synthesis, all of which lead to more effective diagnosis and treatment of illness. [37], [19], [17], [35], [18] give excellent overviews of self-assembly methods and emerging applications. In particular, DNA origami is advancing very rapidly toward its potential as a drug delivery mechanism. For example, [35] and [41] discuss how DNA assembly can improve the treatment of cancer and [16] outlines how self-assembled DNA can be used to help overcome drug resistance in breast cancer cells. Already various labs have produced closed containers self-assembled out of DNA strands which can be made to encapsulate particles, and then their opening deliberately triggered. See [2] and [8] and the HarvarDNAno video [29] which well illustrates both the process and promise of this new technique.

Since modeling this self-assembling process requires designing the component molecular building blocks, which often are modeled through surface meshes, lattice subsets, and other graph-like structures, construction methods developed with concepts from graph theory have resulted in significantly increased efficiency. For example, one recent focus in DNA nanotechnology is the formation of nanotubes which can be modeled using a lattice graph. The rules governing the structure of these nanotubes are not yet well understood, and this naturally offers open problems in the realm of applied graph theory.

1.2 Research Aims and Goals

In this research, we explore the underlying graph-theoretical structure of nanostructure construction and related design strategy problems. Our project focuses on a construction method for self-assembling DNA structures which involves branched junction molecules whose flexible k-arms are double strands of DNA. These arms have cohesive ends which can bond to any other cohesive end with a complementary sequence of bases. We call the k-armed molecule a “tile” and represent it as a vertex of degree k in a graph. We say a collection of tiles (a pot) realizes a graph, $G$, if the collection of tiles constructs the same structure as $G$. Thus, we can investigate design process questions regarding which types of final structures can be constructed from a given pot of tiles.
Inversely, we can also find a pot of tiles that will realize a given target graph. Our group primarily focuses on the latter problem. Along with graph theory and combinatorics, we also use linear algebra to help answer these questions through the “construction matrix” [5]. The goal in answering these questions will be to determine the minimum number of branched junction molecules and their combinatorial structures for self-assembling DNA nanostructures.

The central focus of our research is to find the minimum number of tile types and bond-edge types necessary to construct a target graph \( G \) by modular assembly under a variety of laboratory conditions, and furthermore find explicit pots realizing these minima. This includes creating general design theory and finding accurate bounds for the number of tile types and bond-edge types for a variety of new graphs. Thus, the fundamental questions for this research include:

- Given a target graph, \( G \), what is the minimum number of tile types which need to be designed to create the graph? We denote this minimum number by \( T(G) \).
- Given a target graph, what is \( B(G) \), the minimum number of bond types needed?
- Given a target graph, what is the combinatorial structure of the molecules in a minimum set which realizes the target graph?

Questions regarding design strategies for realizing a target graph are considered under three different scenarios:

**Scenario 1:** A graph with a fewer number of vertices than the target graph may be realized from a pot of tiles.

**Scenario 2:** A graph on the same number of vertices, but not isomorphic to the target graph may be realized from a given pot of tiles. However, no graph on fewer vertices may be realized.

**Scenario 3:** No graph on fewer vertices nor non-isomorphic graphs on the same number of vertices as the target graph may be realized from a given pot of tiles.

Our goal is to provide at least tight bounds, and ideally provably optimal closed form solutions, under each scenario. My faculty research team has recently submitted a paper discussing partial results for grid graphs, nanotubes, and platonic solids. We ultimately intend to build on our results to obtain provably optimal closed form solutions for Scenarios 1 and 2 for each of the graph types mentioned. For Scenario 3, we aim to first find and then prove tight bounds on numbers of tile and bond-edge types. We also plan to develop obstruction theorems which can help provide support for laboratories creating these nanostructures. Specifically, we hope to find design rules to help the laboratories avoid creating smaller graphs in some of the more restrictive scenarios. The problem of finding optimal pots of tiles leads to one of our other goals for our research which is to create tools to help find and check pots of target graphs for different scenarios. We have already made progress towards this goal. My faculty team has already written two Maple programs to help us produce and check pots for Scenario 2. One program is designed to check the minimum size graphs a given pot of tiles will construct in Scenario 2. Specifically this program deals with the case in which a free variable appears in the solution to the system represented by the construction matrix.
Our second program produces all possible pots for \( k \)-regular graphs on \( n \) vertices using \( b \) bond-edge types. The program confirms that no smaller graphs can be realized from the pots produced.

1.3 Methodology

The introduction of a graph-theoretical formalization for exploring the combinatorial properties within the self-assembly of branched junction molecules was first introduced by Jonoska et al. in [12]. The centers of these branched molecules form the vertices of the constructed complex. One arm of the molecule extends longer than the other and its end forms a sticky (or cohesive) end-type which can bond with other arms with the corresponding complementary Watson-Crick Bases. See Figures 2 and 1.

To represent these \( k \)-armed molecules and sticky end types, we will use a vertex of degree \( k \) like the ones in Figure 1. We call these \( k \)-armed molecules “tiles.” We represent the complementary sticky-ends or bond-edges of these tiles using letter labels as in Figures 2 and 1. The letters \( a, b, c, \hat{a}, \hat{b}, \hat{c}, \) etc. represent the un-adjoined arms sticking off of the \( k \)-armed molecules. For example, bond end types \( a \) and \( \hat{a} \) represent complementary sequences of bases as in Figure 2. Thus \( a \) and \( \hat{a} \) could self assemble in order to form a bond-edge. Thus we can combinatorially represent a \( k \)-armed branched-junction molecule with bond-end types \( a_1, \ldots, a_k \) using a tile \( t = (a_1, \ldots, a_k) \).

Figure 1: 3-armed branched junction molecule (left) with example tile representation (right)

We assume these tiles have arms which can freely move and are long enough and flexible enough to reach around and connect with complementary cohesive end-types without restriction. These “flexible tiles” were first introduced in [11]. Thus we define a tile type \( t \) to be a flexible-armed branched junction molecule described by a set of cohesive-end types. We call a set of tiles, a pot. Figure 3 gives an example of a pot with 4 different tile types.

Figure 2: Representing the complementary cohesive end types

Figure 3: A pot of 4 tile types.
Theoretically we assume that once a tile type is made, a pot with that tile type can have “infinitely many” of those tile types. Once we have a pot, those tiles are able to self-assemble into complexes either complete or incomplete as seen in Figure 4. Labs generally want to create complete complexes.

![Figure 4: Complete vs Incomplete Complex](image)

Linear algebra and programming can also be used to help answer these questions. In particular, the construction matrix can be used to help determine whether a pot might generate graphs smaller than the target graph. [5, 12]. Given a pot $P$ of tiles, in order for a complete complex to be formed and no smaller graph to be formed, the proportions of different tile types must satisfy a system of equations which capture the requirements outlined in Item 2 of Proposition 2 in [5]. We often represent this system of equations with an augmented matrix called the construction matrix, denoted $M(P)$. If the solution to the construction matrix provides a unique solution, then no smaller complexes can be made from the pot that determines $M(P)$. Thus, the construction matrix is a useful way to determine whether pots can generate graphs smaller than the target graph.

### 1.4 Mentorship Plan

To help keep the students engaged in the research, the students will meet at least three times a week with Dr. Harsy in addition to attending the SURE seminars and the weekly Math Research Seminars. During the Math Research Seminars, students will present their current progress on their research to the other students and faculty working on math research. This allows them to practice presenting and gives them a chance to get feedback from other mathematicians. Most students present bi-weekly. Students will also be expected to keep a research journal and complete a final write-up and research report/paper by the end of the summer. The schedules below provide details and timelines for the students and will help keep them accountable and organized with completing their work. Many of the students who have worked on this research have presented their results at local, regional, or national mathematics conferences and I would expect the students in this project to present at similar conferences over the following year.

Students working on this project will design optimal pots for different families of graphs. 16 Lewis students have worked on this project so far and the majority of these students have been determining and proving optimal designs for families of bipartite graphs and lattice-based graphs in Scenarios 1 and 2. Two students have been working on computer programs to help with the design process. Depending on the SURE student’s interests, the student may continue research already done by past students, explore new graphs, or work to develop additional mathematical software to be used directly by the laboratories creating the nanostructures. Ideally, we would like students to explore lattice-based graphs or graphs that grow in more than one way since these will be most applicable
for designs of drug delivery containers. The possible projects for the research (depending on the student’s interest and background) are outlined below:

- **Project 1:** Compute $T(G), B(G)$, and find explicit pots of tiles for families of graphs with a lattice substructure such as Mongolian Tent Graphs, Sierpinski Carpet Graphs, Ladder graphs, Web, Helm, Cone, Sunlet, etc.

- **Project 2:** Continue the progress from past student research by computing $T(G), B(G)$, and finding explicit pots of tiles for families of graphs which grow in multiple ways like fan graphs, stacked book graphs, Mongolian Tent Graphs, Ladder Graphs, etc.

- **Project 3:** Compute $T(G), B(G)$, and find explicit pots of tiles for families of $k$-regular graphs such as Crown Graphs, Cage Graphs, and $n$-circular graphs. Additionally, students can explore relationships between pots of tiles for $k$-regular graphs for a fixed value of $k$.

- **Project 4:** Expand and improve the efficiency of current software used to produce and check pots for regular graphs, in addition to computing pots of tiles for non-regular graphs.

Regardless of the project the student chooses, I have a particular way which I prepare students for this research. Below is a detailed plan for the first two weeks (6 meetings) of the research which provides background and allows students to learn the typical techniques, notation, and logic used in this research.

**Day 1: Introduction to research topic and notation:** Students will read *Using DNA self-assembly design strategies to motivate graph theory concepts* pages 96–101 along with the first three chapters of Alan Tucker’s *Applied Combinatorics*.


**Day 3: Scenario 2 Techniques and Introduction to Using the Construction Matrix:** Students will continue reading *Minimal tile and bond-edge types for self-assembling DNA graphs*, specially pages 245–247 and the Scenario 2 section on pages 250–253. Students will start creating and justifying optimal tiling for known graphs like $C_n, K_n, S_n, W_n$, and $P_n$ in Scenario 2. The student will be introduced to the main linear algebra construct used in this research.

**Day 4: Scenario 2 Techniques Continued:** Students will finish reading *Minimal tile and bond-edge types for self-assembling DNA graphs* pages 254–256, 260–264. Students should have finished creating and justifying optimal tiling for known graphs like $C_n, K_n, S_n, W_n$, and $P_n$ in Scenario 2.

**Day 5: Scenario 3 Techniques:** Students should have finished reading *Minimal tile and bond-edge types for self-assembling DNA graphs* and should start creating and justifying optimal tiling for known graphs like $C_n, K_n, S_n, W_n$, and $P_n$ in Scenario 3.
Day 6: Scenario 3 Techniques Continued: Students should be done creating and justifying optimal tiling for known graphs like $C_n$, $K_n$, $S_n$, $W_n$, and $P_n$ in all 3 scenarios and will decide on which project/graphs they would like to explore on their own.

1.5 Proposed General Timeline with Project Goals

Weeks 1-2: Read the first three chapters of Alan Tucker’s *Applied Combinatorics* along with some basic papers including [4] and [5] which introduce the project. Students should work on exercises and confirm results from these papers.

Weeks 3-7: Develop design strategies to create minimum pots for different bipartite, tripartite, or lattice-based graphs in Scenarios 1, 2, and 3. Formalize the results by constructing proofs or counter examples.

Weeks 8-9: Finalize and summarize results, attend and present at MAA MathFest.

Week 10: Finish main draft of paper and prepare for SURE presentation

1.5.1 Timeline for Student Write-Up/Paper Task List:

Week 1: Do a literary search for the current research in Modeling DNA Self-Assembly. Read [4] and [5] which introduce the project. Complete a typed 1-3 paragraph synopsis of what might be useful in the articles.

Week 2: Work with Dr. Harsy to decide on the focus of their research and the point of their paper. At this point they should have a title page and the TeX template set up. This is just a working title and can be changed.

Week 3: Make a concept map of the ideas for the paper and how they interconnect. Decide how to thread your storyline through this map.

Week 4: First draft of definitions and background terminology and notation, three examples, or other foundation materials should be added to the report.

Week 5: Complete further work on definitions and outline of content. Start including examples or data or initial results.

Week 6: Write a draft of the introduction. This will incorporate material from the paragraphs you wrote during the literary search. Continue adding supporting materials. Start working on any figures you want to add to the paper.

Week 7: Draft a conclusion and typeset. Insert figures, review expository structure. If possible, write a 15 min talk with slides, and use what you learn from the talk to improve the organization, figures, and exposition of the paper.

Week 8: Do an overall editing pass to complete the first draft. Continue to add further results if needed.

Week 9: Have someone give you some feedback like Dr. Harsy, Dr. Meyer, or Dr. Stephenson. Having someone outside of your project is often helpful at this stage. Finalize SURE presentation.

Week 10: Edit the second draft. Put it away for at least a week, edit again (have someone give feedback again). Do final editing and then possibly submit to a journal.
References


4 Criteria for Student Applicants

This research can be approached with little mathematical background. It involves basic graph theory and depending on the student’s background and interests may also involve programming, computer graphics, art, biology, and geometry.
