3DNA Printer: A Tool for Automated DNA Origami

Amay Agrawal, Birva Patel, Dixita Limbachiya and Manish K. Gupta
Laboratory of Natural Information Processing
Dhirubhai Ambani Institute of Information and Communication Technology Gandhinagar, Gujarat, 382007 India
Email: amayagrawal22@gmail.com, birva1@gmail.com, dlimbachya@acm.org, mankg@computer.org

Abstract—In the last two decades, DNA self-assembly has grown into a major area of research attracting people from diverse background. It has numerous potential applications such as targeted drug delivery, artificial photosynthesis etc. In the last decade, another area received wide attention known as DNA origami, where using M13 virus and carefully designed staple strands one can fold the DNA into desired 2-D and 3-D shapes. In 2016, a group of researchers at MIT have developed an automated DNA nanostructures strategy and an open source software 'daedalus' based on MATLAB for developing the nanostructures. In this work, we present a truly open source software '3dnaprinter' based on Java (without MATLAB) that can do the same work.

I. INTRODUCTION

Apart from being the blueprint of the life, DNA has been used for different applications [15]. One of the application is to built nanostructures from DNA. DNA consists of four main bases namely A (Adenine), T (Tyhmine), G (Guanine) and C (Cytosine), in which G and C while A and T are complementary bases and they form double helix when they come together via hybridization. In 1980s, this intrinsic property was utilized by Ned Seeman and his collaborators to construct many interesting DNA structures at the nanoscale by trial and error such as DNA cube, DNA octahedron, DNA double crossover (DX) molecules etc [10], which gave birth to a field called structural DNA nanotechnology. In 1995 Erik Winfree explained that the self-assembly of DNA is Turing-universal [16]. That means, in principle, rather than trial and error one can systematically design any arbitrary shape with DNA. In a seminal paper in 2006, Rothemund introduced a method called DNA origami [9], which is essentially the art of folding using DNA. By using all these self-assembly techniques one can design various interesting shapes such as smiley face, star, map of countries etc. In DNA Origami method, one can take a long single "scaffold" strand, that on hybridization at specific positions using "staple" strands results into the desired nanostructure.Since then, DNA nanostructures were built using distinct approaches as DNA tiles [13], DNA bricks [6]. The systematic approach was proposed by Peng Yin et al. to construct any arbitrary 2-D shape by DNA using DNA Tiles as the basic building block. Based on this concept, a software DNA Pen was developed [3]. Scaling up this 2D model for 3D shapes, Peng Yin and his colleagues proposed the idea of DNA bricks which works like lego modeling to build the 3D structures. A software 3DNA is developed on this concept which allows designing of the 3D shapes by sculpting of DNA nanostructures [4]. DNA nanostructures can be used in applications such as biosensing [11], triggered drug delivery [7], enzyme cascades [14], bio-molecular analysis platforms [2] etc.

In this work, we introduced the software called 3DNAprinter to design the 3D structures using the systematic algorithmic top-down approach. This software is motivated from the work of Veneziano et al [12]. A software called DAEDALUS has been developed on this idea [12]. The main difference between DAEDALUS and 3DNAprinter software, is that our software is an open source implemented in java while DAEDALUS is an open source implemented in MATLAB, which is not an open source even though it is free to use. We have slightly modified the algorithm used in DAEDALUS for efficiency. This software basically takes in the .obj file of any 3d shape (current version handles platonic solids) and converts it automatically into corresponding DNA sequences using the structured algorithmic self assembly. The 3 main steps employed in formation of structure are spanning tree generation, Eulerian circuit formation and staples addition.

The paper is organized as follows. Section 2 contains an algorithm used to build the structure. Section 3 discuss the GUI of our software. Section 4 contains functionality and work-flow of our software. Section 5 contains few examples that have been build using our software. Section 6 provides the link from where our software is available. Section 7 concludes the paper with some general remarks.

II. ALGORITHM

In this section, we recall the algorithm used in [12] for constructing 3D DNA origami. The main purpose of this algorithm is to model and synthesize a DNA scaffold and its staples for any given 3-D object using the top-down approach. For this purpose, one can visualize any 3-D object into its corresponding 3-D graph. The algorithm firstly generates a DNA scaffolding route which is followed by generation of staples which pull the structure together to give the required 3-D shape. The steps of the algorithm are schematically represented in Figure

A. Generating the spanning tree

For the algorithm to successfully route the 3-D structure, the basic requirement is that the Eulerian Circuit exists. An Eulerian Circuit is guaranteed if the degrees of the vertices are even and for that purpose we have decided to use 2 duplexes per edge. There are a number of routing solutions which deliver an Eulerian Circuit but not all of them will result in an
effective scaffold routing. Thus to produce an effective scaffold route we have employed the Prims algorithm for the minimum spanning tree. As the 3-D objects have equally weighted edges, any of the minimum spanning tree algorithms like Prim’s, Kruskal’s, DFS, BFS etc. could be used. But as we know that branching tree self-assemble more reliably than linear trees \[8\], hence we have used Prim’s algorithm which gives breadth first search spanning tree with most branches. We know that the crossovers can be used to strengthen the 3-D structure \[1\]. Crossover can occur in staple strands or in scaffold strands. According to our algorithm, each edge can correspond to either zero or one scaffold crossover. The edges that are part of the spanning tree are assigned a zero scaffold crossover and the remaining edges are assigned one scaffold crossover. This can be seen in the Figure \[1\]b.

B. Adding pseudo-nodes and scaffold routing

On formation of spanning tree, the graph is converted to an Eulerian circuit. This is done by splitting the non-spanning tree edges into two halves by adding a pair of pseudo-nodes and keeping all the spanning tree edges intact. Depending on the degree of the vertex, a set of pseudo-vertices are added to replace that vertex node. In consequence to this, each pseudo-vertex of degree N has N edges emerging from this and N faces between them. Each pseudo-node is placed such that it joins two bordering edges of a face and disconnects that face from other edges. This forms the Eulerian Circuit of the 3-D object through which the scaffold will route.

The circuit can be routed from two different directions, either in clockwise or anticlockwise direction. But for the stability purposes \[5\], we will route in the anticlockwise direction. Thus now we have a directed Eulerian graph. This can be seen in Figure \[1\]c.

C. Sequence generation and adding staples

From the Eulerian circuit constructed above, we can now route the DNA strand across the scaffold. As edge should be a multiple of 10.5 bps, we have employed to use 52 bps for the spanning tree edges while 26 for the non-spanning tree cut edges.

Our algorithm used 3 different kinds of staples: vertex staples, zero scaffold crossover edge staples and one scaffold crossover edge staples. These staples can be easily created by taking the complimentary base pairs of the scaffolded DNA strand. We have created these staples using two stack algorithm.

Vertex staples can be generated by traversing the Eulerian circuit using two stack algorithm. For each edge that is a part of vertex staple, we use 10 bps which can be from either 3’ or 5’ end that is complementary to the corresponding scaffold base pair.

For edge staples with one scaffold crossover, two 31 bps - 32 bps staples are placed across the crossover with 15 bps - 16 bps on either side of crossover to provide strong binding. While the remaining 20 bps - 21 bps corresponding to that edge has been already used in forming the vertex staples.

In edges with zero scaffold crossover, it has 42-nts staple with a staple crossover at 21 bps. The remaining 20 bps - 22 bps are employed to make another C-shaped staple.

III. Graphical User Interface

The GUI of the 3DNAprinter has been developed for the users to easily upload any 3D .obj file of any shape and get the corresponding DNA sequences of the shape to construct it at a nanoscale. Schematic representation of GUI is described in Figure \[3\].

A. Importing a file and Creating Polyhedra mesh

As shown in the Figure \[4\] one can find this button on the top left corner where one can import any 3D .obj file and the
corresponding 3D object can be obtained as shown in Figure 1.a

B. Spanning Tree Generation of Polyhedra mesh

As shown in the Figure 5 one can find this button on the top right corner where one can just click the button and the spanning tree corresponding to the 3D structure can be obtained as shown in Figure 1.b

C. Forming the Eulerian circuit for the scaffold path

As shown in the Figure 6 one can find this button on the top right corner where one can just click the button and see the Eulerian circuit which will then be used to route the scaffold along it. It is shown in Figure 1.c

D. Generating the DNA sequences corresponding to the structure

As shown in the Figure 7 one can find this button on the top right corner where one can just click the button and the DNA sequences corresponding to the imported 3D structure will be saved in excel file named Output_Sequences.xls which can be found in the sequences folder.

IV. FUNCTIONALITY AND WORKFLOW

The main aim of this subsection is to provide an overview of the working and functionality of the software. The key features of the software are its importing of the 3-D object files and output modules which include saving sequences corresponding to the 3-D object into the files.

A. Creating a polyhedra mesh

A new 3-D object file can be easily imported from the import file option. This 3-D object file (.obj extension) can be easily created in Auto-Cad, Maya etc. The user can then have a 3-D view of the corresponding object which is formulated by java3d libraries. The constructed polyhedron mesh can be seen in the figure 1.a

B. Spanning tree generation

In this step, users can view the spanning tree of the corresponding 3-D object, which is made using prims algorithm. The edges that are a part of the spanning tree has been shown in red color which can be seen in figure 1.b

C. Scaffold routing path generation

From the spanning tree, an Eulerian path is formulated through which the DNA strand will route the 3-D structure. This route is made by splitting the vertices into the pseudo-vertices corresponding to its degree and cutting the non-spanning tree edges. The figure 1.c shows the entire Eulerian scaffold routing path where the starting edge of the Euler circuit is shown in red and the next edge in blue.

D. Generating output sequences

The set of sequences generated by the software are stored in a Excel sheet (.xls extension). The file contains the vertex-to-vertex DNA scaffold sequences, vertex staple sequences, edge staple sequences and cut-edge staple sequences corresponding to the 3-D object.
**E. Reset module**

The user can reset the canvas and import a new 3-D object file corresponding to a new structure.

**F. Estimator module**

This module can be used to calculate the amount based on the number of bps.

**V. EXAMPLES**

Using 3DNA Printer, one can model various 3-D structures of varying degrees like a tetrahedron, octahedron, icosahedron etc. By default, we are using M13 virus of length 7429 bps to route the scaffold although users can select the DNA strand of their choice. The details of constructing two 3-D structures namely tetrahedron and octahedron using our software is given as follows:

**A. Tetrahedron**

The pictorial representation of the tetrahedron can be seen in figure 8 which is explained below:

- A tetrahedron consists of 4 vertices which each vertex having a degree of 3, as shown in figure 8.a.
- The spanning tree of the tetrahedron is formulated using the prim’s minimum spanning tree algorithm. As seen in the figure 8.b, edges 1-2, 1-3, 1-4 are part of the minimum spanning tree which are represented in red color.
- The non-spanning tree edges i.e edges 2-3, 2-4, 3-4 are now split into two halves and whereas all spanning tree edges are kept intact. Also, each vertex is replaced by 3 pseudo-vertices to form a Eulerian circuit.
- Now after forming the Eulerian circuit, the default M13 is routed in the anticlockwise direction across the scaffold with each edge having 52 bp. As seen in figure 8.c, starting edge is represented in red and the next edge in blue.
- Thus, we require a total of 4 vertex staples, 6 edge staples and 6 cut-edge staples for the formation of tetrahedron.
- A total of 1026 nucleotide bps is required for the formation of tetrahedron.

**B. Octahedron**

The pictorial representation of the tetrahedron can be seen in figure 9 which is explained below:

- An octahedron consists of 6 vertices which each vertex having a degree of 4, as shown in figure 9.a.
- The spanning tree of the octahedron is formulated using the prim’s minimum spanning tree algorithm. As seen in the figure 9.b, edges 1-2, 1-3, 1-4, 1-5, 2-6 are part of the minimum spanning tree which are represented in red color.
- The non-spanning tree edges i.e edges 2-3, 2-5, 3-4, 4-5, 3-6, 4-6, 5-6 are now split into two halves and whereas all spanning tree edges are kept intact. Also, each vertex is replaced by 4 pseudo-vertices to form a Eulerian circuit.
- Now after forming the Eulerian circuit, the default M13 is routed in the anticlockwise direction across the scaffold with each edge having 52 bp. As seen in figure 9.c, starting edge is represented in red and the next edge in blue.
- Thus, we require a total of 6 vertex staples, 10 edge staples and 14 cut-edge staples.
- A total of 2058 nucleotide bps are required for the formation of tetrahedron.

**VI. AVAILABILITY**

This software can easily be downloaded from the following URL: http://www.guptalab.org/3dnaprinter/

**VII. CONCLUSION**

This paper present a Java (without MATLAB) based open source software for developing 3D nanoscale structures automatically by importing a corresponding .obj file which can be obtained from any well known software like Autocad, Maya etc. for any desired 3D structure. Current version of the software designs only Platonic solids. In future, we will release a version which supports any 3D object.
REFERENCES

[1] J. R. Arias-Gonzalez, “Entropy involved in fidelity of dna replication,” PloS one, vol. 7, no. 8, p. e42272, 2012.
[2] M. M.-C. Cheng, G. Cuda, Y. L. Bunimovich, M. Gaspari, J. R. Heath, H. D. Hill, C. A. Mirkin, A. J. Nijdam, R. Terracciano, T. Thundat et al., “Nanotechnologies for biomolecular detection and medical diagnostics,” Current opinion in chemical biology, vol. 10, no. 1, pp. 11–19, 2006.
[3] A. Goyal, D. Limbachiya, S. K. Gupta, F. Joshi, S. Pritmani, A. Sahai, and M. K. Gupta, “Dna pen: A tool for drawing on a molecular canvas,” Poster presented at 19th International Conference on DNA Computing and Molecular Programming, 2013.
[4] S. K. Gupta, F. Joshi, D. Limbachiya, and M. K. Gupta, “3dna: A tool for dna sculpting,” arXiv preprint arXiv:1405.4118, 2014.
[5] Y. He, M. Su, P.-a. Fang, C. Zhang, A. E. Ribbe, W. Jiang, and C. Mao, “On the chirality of self-assembled dna octahedra,” Angewandte Chemie, vol. 122, no. 4, pp. 760–763, 2010.
[6] Y. Ke, L. L. Ong, W. M. Shih, and P. Yin, “Three-dimensional structures self-assembled from DNA bricks,” Science, vol. 338, no. 6111, pp. 1177–1183, 2012.
[7] V. Linko, A. Ora, and M. A. Kostiainen, “Dna nanostructures as smart drug-delivery vehicles and molecular devices,” Trends in biotechnology, vol. 33, no. 10, pp. 586–594, 2015.
[8] S. Pandey, M. Ewing, A. Kunas, N. Nguyen, D. H. Gracias, and G. Menon, “Algorithmic design of self-folding polyhedra,” Proceedings of the National Academy of Sciences, vol. 108, no. 50, pp. 19885–19890, 2011.
[9] P. W. Rothemund, “Folding DNA to create nanoscale shapes and patterns,” Nature, vol. 440, no. 7082, pp. 297–302, 2006.
[10] N. C. Seeman, “Nucleic acid junctions and lattices,” Journal of theoretical biology, vol. 99, no. 2, pp. 237–247, 1982.
[11] Y.-T. Tai, Y.-W. Chang, M.-Y. Lin, Y.-S. Yang, and Y.-T. Huang, “A high-sensitivity cnos-compatible immunosensor based on the hpr/tmb/streptavidin system,” in Sensing Technology, 2008. ICST 2008. 3rd International Conference on. IEEE, 2008, pp. 297–300.
[12] R. Veneziano, S. Ratanalert, K. Zhang, F. Zhang, H. Yan, W. Chiu, and M. Bathe, “Designer nanoscale dna assemblies programmed from the top down,” Science, vol. 352, no. 6293, pp. 1534–1534, 2016.
[13] B. Wei, M. Dai, and P. Yin, “Complex shapes self-assembled from single-stranded DNA tiles,” Nature, vol. 485, no. 7400, pp. 623–626, 2012.
[14] O. I. Wilner, Y. Weizmann, R. Gill, O. Lioubashevski, R. Freeman, and I. Willner, “Enzyme cascades activated on topologically programmed dna scaffolds,” Nature nanotechnology, vol. 4, no. 4, pp. 249–254, 2009.
[15] O. I. Wilner and I. Willner, “Functionalized dna nanostructures,” Chemical reviews, vol. 112, no. 4, pp. 2528–2556, 2012.
[16] E. Winfree, “Algorithmic self-assembly of dna,” Ph.D. dissertation, California Institute of Technology, 1998.