Tiamat: A Three-Dimensional Editing Tool for Complex DNA Structures

Sean Williams[1], Kyle Lund[2,4], Chenxiang Lin[2,4], Peter Wonka[3], Stuart Lindsay[2,4,5], Hao Yan[2,4]

[1]Department of Computer Science
University of California, Davis
Davis, CA, 95616
[2]Center for Single Molecule Biophysics, The Biodesign Institute
Arizona State University
Tempe, AZ, 85287
[3]School of Computing and Informatics
Arizona State University
Tempe, AZ, 85287
[4]Department of Chemistry and Biology
Arizona State University
Tempe, AZ, 85287
[5]Department of Physics and Astronomy
Arizona State University
Tempe, AZ, 85287

Abstract

We present the development of a new graphical user interface driven molecular modeling, editing and visualization tool called Tiamat. Tiamat addresses the challenge of how to efficiently model large and complex DNA nanostructures. We describe the three major components of our system. First, we discuss design guidelines and data structures that form the basis of flexible and large-scale editing. Second, we explain a semi-automatic sequence generator that combines user input with random sequence generation to efficiently label the molecules in the DNA structure. Third, we outline the visualization techniques including a simplification algorithm that are used to render large designs. The results demonstrate how Tiamat was used to generate large and complex designs.

Keywords: Structural DNA nanotechnology, DNA tile, 3D DNA structure display, DNA sequence design

1 Introduction

The concept of the immobile branched DNA junctions [1], proposed by Seeman in 1982, led to the use of DNA as a tool for constructing nanoscale arrays and objects [2, 3]. The designs of most DNA nanoarrays are constrained by the helical repeat in DNA of 10.5 bases. Designing 1-dimensional (1D), 2-dimensional (2D) and 3-dimensional (3D) structures using a simple graphics program is possible using multiples of 10.5 bases in designing the crossover points. When trying to make very complex structures, a DNA modeling program is needed to see precisely how the geometrical design is going to fold. Another key aspect to designing DNA nanostructures is the generation of a random sequence with minimized sequence symmetry where there is no secondary structure that can be formed by the DNA.

Already in 1985, Seeman produced a command line driven Fortran program for creating relatively simplistic drawings of DNA structures [4]. The lack of a graphical interface in the program made the direct observation of the 3D aspect of the DNA structure difficult. Alternatively, the jacks and straw modeling of DNA was used for making simple structures, but as the structure increased in complexity this became very difficult to handle [5]. Recently, Birac and Seeman developed GIDEON, a 3D DNA rendering program [6]. GIDEON is a useful program that allows a user to design complex and precise DNA nanostructures. The program makes it possible to see the designed structure in top, side and front views. GIDEON has the ability to predict the actual distortions that will occur to the DNA structure. Overall, GIDEON is a great tool for the design of DNA structures. In this paper we build on this successful previous work while addressing some of the limitations that make it difficult to efficiently generate large designs. The first limitation is that the sequence generation is not integrated with the modeler.

The generation of DNA sequences has been predominantly prepared using a Fortran program called SEQUIN [7]. SEQUIN has the ability to join DNA helices together and then generate a group of sequences one at a time. The joining of DNA helices together requires the user to sketch a 2D line drawing of the structure. The structure is then put into the program by designating arm lengths and linking the arms together. Once the structure has been implemented in the program the user can generate the sequence for the structure by comparing new sequences to sequences that already exist in the structure to avoid sequence homology. The sequence generator is not random and the user has to decide if they like the sequence or if they want the computer to generate another sequence. Also, it is cumbersome to input very large structures into a separate, command-line tool.

The second limitation of previous work is that the design of data structures and visualization strategies were not flexible enough to allow for large and complex designs. The large number of molecules in a single design might become confusing for the designer and can also overwhelm the graphics hardware. In this paper we present our modeling program Tiamat to overcome these limitations. Tiamat is a 3D modeling program designed specifically for the creation of large and complex DNA structures. This paper contains three major contributions. First, a design interface focused on flexibility in modeling structures. The structures allow the modeling of general designs. Second, an integrated random sequence generator that checks sequences directly against the provided model to prevent the formation of secondary structures. Third, a fast and dynamic level of detail algorithm to simplify display of very large structures while still displaying important information.

2 Methodology

This section will describe the important design and implementation considerations for working with structural DNA, with overall structure sizes ranging from dozens to tens of thousands of bases. In section 2.1, we will discuss the data structures usable for structural DNA modeling and their impact on interactivity. In section 2.2, we will discuss random sequence generation directly applied to a DNA model. In section 2.3, we will discuss a dynamic level of detail algorithm for very large DNA structures.

2.1 Data Structures

In this section, we will discuss various approaches to storing and interacting with the data that describes the components of a DNA structure. A natural starting place is an undirected graph [8], a data structure in which the data is stored in vertices that can be arbitrarily connected by edges, with the caveat that if an edge exists between vertices A and B, an edge also exists between B and A. With this in mind, there exists a more important question of what a vertex represents.

The high level approach is to have each vertex represent a DNA helix. Each vertex would store the five-prime sequence (the three-prime sequence can be inferred to be the complement of the five-prime sequence), with four potential edges: the top and bottom of the five-prime strand, and the top and bottom of the three-prime strand. By dividing a helix – converting it into two helices with the bottom of the top helix connected to the top of the bottom helix – junctions to other helices could be added with little trouble. In this approach, helix-centric operations are easy, such as calculating the physical deformation caused by junctions. However, base-centric operations, such as the creation of arbitrarily-shaped probes, are more difficult.

The low level approach is to have each vertex represent a base. Each vertex would store its own Watson-Crick type (e.g. cytosine), and its edges would connect to its neighboring bases: the base up the strand, the base down the strand, and its paired base. With this model, the creation of junctions and other deviations from a standard helix are accomplished by reassigning the edges of the involved bases. This model makes base-centric operations easy, while making helix-centric operations difficult.

There are benefits and drawbacks to both choices, so the strengths of each implementation are based on the necessity of its associated features. In a base-centric design some elements of higher level structure can be inferred from connectivity; that is, by traversing along up and down edges one can determine all the bases that compose one strand. However, if two helices are linked by a junction, then a traversal of all edges – which finds the entire helix a base belongs to – will treat the linked helices as one. In this sense, a base-centric design throws away most high level information, making helix-centric operations virtually impossible.

The base-centric and helix-centric designs each have one associated feature under consideration. The helix-centric feature desired was a calculation of the physical deformation a structure would undergo given a configuration of junctions. This stems from the physical requirement that five-prime and three-prime connections of bases all be the same length, since they must all be composed of the same phosphate chains. An algorithm that could reasonably solve this problem would have to perform some sequence of rotations and translations on helices to align them into a valid arrangement; this algorithm cannot be done without helix-centric information.

The base-centric feature desired was a method for creating arbitrarily shaped strands. If a strand is represented in data only as a sequence, its graphical representation derives from well-established biological rules: a double helix with a known rotation per base pair, distance between base pairs, and so on. Creating an arbitrarily shaped strand – for example, a straight line – cannot be done directly, and would at best require a hybrid implementation in which arbitrary strands are base-centric and standard helices are helix-centric. The primary drawback of such a hybrid approach is that it would be too complicated for any straightforward implementation, effectively crippling its usability.

The final decision was ultimately reached by a desire for functionality over aesthetics. That is, the most important use for a structural DNA modeling tool is to create structures to be reproduced in real life by ordering the strands that will create the structure via self-assembly. Calculating the physical distortions a structure will undergo in Tiamat has relatively little theoretical value, while being given the maximum flexibility in modeling decisions increases the range of structures Tiamat can be used to design. Thus, Tiamat employs a base-centric representation of the structure.

2.2 Sequence Generation

In this section, we will discuss the generation of random sequences directly applied to a DNA structure. Self-assembly dictates that, if the sequences of each strand are assigned correctly, then those strands would assemble into the original structure. However, the sequences must be chosen carefully to avoid the formation of secondary structures, in which unintended Watson-Crick bonds occur and destroy part or all of a structure. To prevent secondary structure formation, Tiamat recognizes three constraints that can be applied to an otherwise randomly generated sequence: unique sequence limit, repetition limit, and GC percentage.

The unique sequence limit refers to the shortest subsequence that must appear only once in the structure. For example, with a unique sequence limit of 5, if a strand contains the sequence ATGACT, then ATGAC and TGACT may not appear anywhere else in the structure (for example, the sequence ATGACC cannot appear anywhere else), though sequences containing ATGA, TGAC, and GACT may appear elsewhere (for example, ATGAGT may appear somewhere else).

The repetition limit refers to the longest sequence of bases that can all be the same. For example, if the repetition limit is 5, then CCCCC may not appear in the structure, though CCCCA or TCCCC can.

The GC percentage refers to the minimum percent of bases in the structure that must be either guanine or cytosine. This places no upper limit on the number of guanine or cytosine that can occur, and for reasons that will be explained below, the results will tend toward more occurrences of guanine and cytosine than specified.

Generating a valid sequence is a discrete optimization problem [9, 10], in which all constraints must be satisfied but any solution that fulfills all the constraints is valid. Exhaustive search approaches to this problem are not feasible, as the size of the search space can easily be shown to be 4^N. That is, an algorithm based on assigning sequences in order until one turns out to meet all the criteria will, for large problem instances, take years to solve. There is a characteristic of this problem that makes it solvable in a short amount of time, however: the size of the set of valid configurations is relatively large with respect to the size of the set of possible configurations. It is therefore reasonable to use a stochastic algorithm to solve this problem.

Our algorithm is inspired by sampling algorithms such as Metropolis-Hastings and Simulated Annealing. We start by assigning all bases a random type. Then, for each base, verify that it meets all of the constraints. If a base is found to violate a constraint, randomly change one of the offending bases. We design a fitness function of the design that includes all constraints and make a random decision on keeping or discarding a change based on changes in the fitness function. The problem with this algorithm is that it lacks a stopping condition; if the constraints are too tight for a solution to be possible, this algorithm will continue forever. A rather inelegant but effective solution to this problem is to note that the algorithm will find a valid solution very quickly if one exists, and to simply put a timeout on the algorithm. That is, if a solution is not found within some number of seconds, to assume a solution does not exist and report the failure to the user. It should also be noted that the minimum valid strength of the constraints is governed by the size of the structure; a very large structure has more connected bases than a very small structure, and must therefore allow more repetitions and longer repeating subsequences in order for a valid solution to exist.

It should also be noted that, in order to reduce the execution time of the algorithm, the random assignment of bases should take into account the GC percentage constraint. For example, if it is required that 70% of bases be guanine or cytosine, it is counterproductive to assign 50% of bases a guanine or cytosine type. However, a sequence of randomly generated numbers lying on a given mean will form a Gaussian distribution around that mean; thus, in order to ensure a solution is more likely to be found, the percentage of guanines and cytosines should be greater than the required percentage by at least one standard deviation. The result of this shift is that more bases than required will be assigned guanine or cytosine, but this is a relatively trivial drawback for its performance increase.

2.3 Visualization

In this section, we will discuss a dynamic level of detail algorithm for very large DNA structures. Levels of detail refers to a process for simplifying the geometry of a model in order to accelerate the rendering of the model [11]. This is typically accomplished through coarsening the polygonal mesh, and selecting an amount of coarseness to represent the mesh based on the camera's distance from the mesh. The general level of detail algorithms work directly on 2-manifold smooth surfaces.

Therefore, the first and most obvious approach is to simplify the geometry of the structure as much as possible while maintaining the same visual elements (see Fig. 1). Since it is still desired in the long run to make nice-looking images, it makes sense to create two drawing modes. The simpler mode exists to draw bases as very simple spheres and connect them with lines; this mode looks coarse, but is much faster to draw. The more complex mode draws bases using

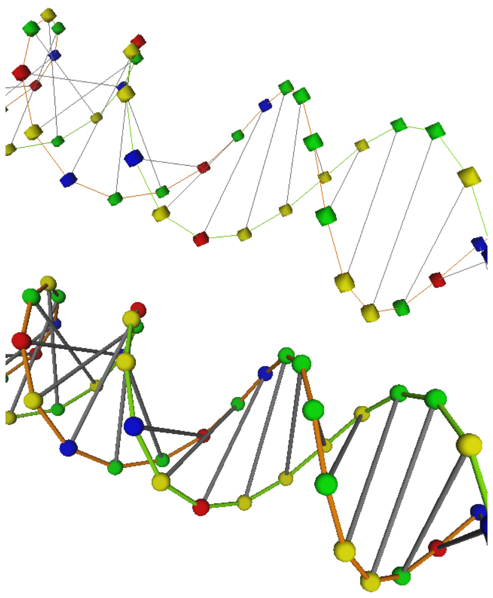
Figure 1: The same helix rendered in two details: edit mode (top) and render mode (bottom)

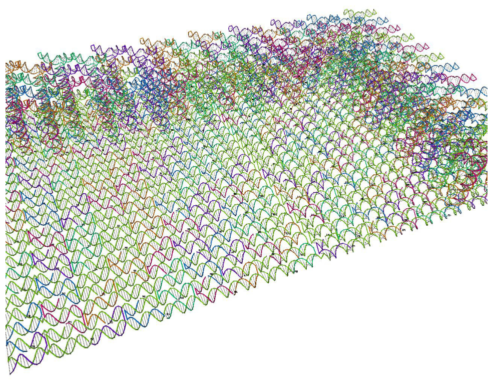
Figure 2: A very large structure; this view is both visually confusing and has a slow frame rate

smooth spheres and connects them using smooth cylinders; this mode looks good, but takes a long time to draw. The draw time difference is significant primarily because editing tools are frustrating to use if user instructions are not carried out at interactive frame rates. With the lower resolution view, even relatively slow computers can be used to create structures, then switched to high resolution drawing only when design of a structure is complete.

Even by coarsening the geometry, a direct rendering will still be very complex for large structures (see Fig. 2). It is difficult to extract useful information from this visualization: while the camera is far away from a helix, the position and orientation of that helix are the only important pieces of information. A better approach is to represent the helices using a proxy geometry, such as straight lines (see Fig. 3, and 4). This is particularly helpful in designing and visualizing large complex DNA nanostructures such as DNA Origami structures recently developed by Rothemund [12].

Converting a helix to a line requires four pieces of information: the direction of the backbone axis, the origin of the backbone axis, the height of the helix, and the bounds at which a helix remains simple (e.g. has no junctions or sharp turns).

The information about a helix is thrown away once the helix is created; see section 2.1 for a detailed discussion on this topic. To reverse engineer the helix direction, we take the position

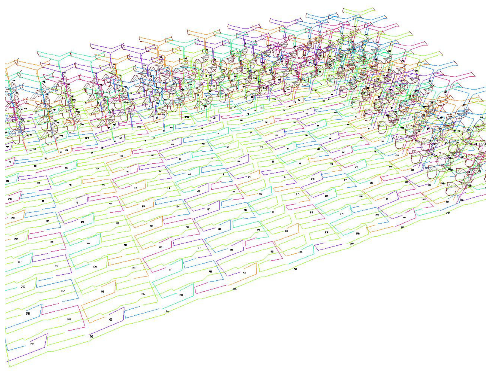

Figure 3: A very large structure with some helices reduced to single lines

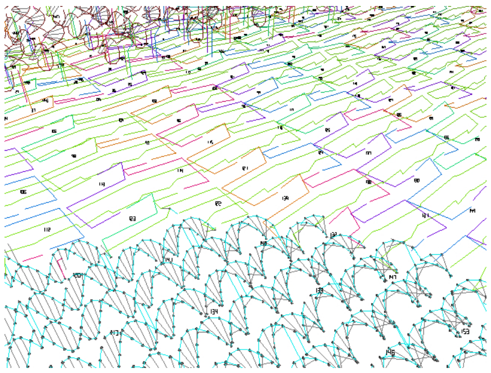

Figure 4: The cutoff between level of detail zones

of the twenty-first base minus the position of the first base, and normalize that. The backbone direction can also be approximated as the average of the vectors from the first to the tenth and the first to the eleventh base, though this approximation is only employed for helices shorter than twenty-one bases. This can finally be applied to even shorter, eight-base groups by using a ratio of approximately 3:1.

To determine the origin of the backbone axis, one of the bases from the second pair is projected into the plane of the first pair; these three points now uniquely define a circle. If the three points are specified as $\mathbf{P}_1$, $\mathbf{P}_2$, and $\mathbf{P}_3$, the center of that circle, which is also the center of the helix, is defined by (4). To determine the height of the cylinder, the same calculation is done for the last base pair in the helix; the length of the vector pointing from the first center point to the last center point is the height of the helix.

$$\alpha = \frac{|\mathbf{P}_2 - \mathbf{P}_3|^2(\mathbf{P}_1 - \mathbf{P}_2) \cdot (\mathbf{P}_1 - \mathbf{P}_3)}{2|(\mathbf{P}_1 - \mathbf{P}_2) \times (\mathbf{P}_2 - \mathbf{P}_3)|^2} \tag{1}$$

$$\beta = \frac{|\mathbf{P}_1 - \mathbf{P}_3|^2(\mathbf{P}_2 - \mathbf{P}_1) \cdot (\mathbf{P}_2 - \mathbf{P}_3)}{2|(\mathbf{P}_1 - \mathbf{P}_2) \times (\mathbf{P}_2 - \mathbf{P}_3)|^2} \tag{2}$$

$$\gamma = \frac{|\mathbf{P}_1 - \mathbf{P}_3|^2(\mathbf{P}_2 - \mathbf{P}_1) \cdot (\mathbf{P}_2 - \mathbf{P}_3)}{2|(\mathbf{P}_1 - \mathbf{P}_2) \times (\mathbf{P}_2 - \mathbf{P}_3)|^2} \tag{3}$$

$$\mathbf{C} = \alpha\mathbf{P}_1 + \beta\mathbf{P}_2 + \gamma\mathbf{P}_3 \tag{4}$$

The final and most difficult challenge is determining the bounds of a simple helix (see Fig. 5). For a simple helix, one can traverse from a base back to itself by going across, down, across, and down. (Note that, since the two strands of a helix are anti-parallel, the down direction of one strand is the up direction of its complimentary strand. Thus, going down one turn on a

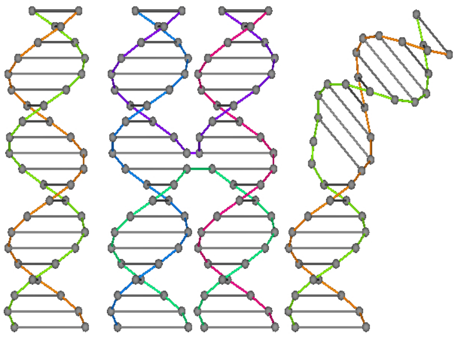

Figure 5: Three cases for circular traversal: first, a simple helix; second, a junction; third, a sharp turn; of these cases, the first can be represented by one cylinder, the second by four cylinders, and the third by two cylinders

strand then down on its compliment will traverse in a circle.) In the case of a junction, going across and down sends a traversal into the other helix, so going across and down again will find a base in a different helix. In the case of sharp turns, in order to keep a physically possible shape, "filler" bases must be added on the wide side of the turn, so again a circular traversal will not return to its starting point. Given all this, a section of a helix can be simplified to one line if and only if a circular traversal is possible for each base on that segment.

Given all this information, during the drawing routine, if a base is far enough away from the camera, traverse up and down from that base to find either where the helix stops being simple (this can be precalculated) or where the simple helix gets too close to the camera, draw a properly colored line between the two ends, and mark that all bases in between have been drawn.

3 Results

To validate that Tiamat is a functional tool and useful in structural DNA nanotechnology a previously designed 4x4 DNA tile[3e] was re-modeled using Tiamat (see supplemental information for strand sequences generated). The design was to use a single 4x4 DNA tile to grow into a large 2D array with the sequence generated through Tiamat. The structure was made according to the design by Yan et al. [3e] incorporating four 4-arm branched junctions and the adjoining junctions are orthogonal to each other and each end has a five base sticky-end overhang. The design uses the corrugated design which allows for the tiles to form large 2D arrays instead of folding on themselves and forming tube structures. Figure 6A is a Tiamat output that shows the design of the 4x4 tile, and Figure 6B is the joining of four tiles to make the beginning stages of a 2D array.

We used Atomic Force Microscopy (AFM) and native polyacrylamide gel electrophoresis (PAGE) to verify if the 4x4 structure and the self-assembled arrays designed by Tiamat formed correctly. Native PAGE (see Figure 7) was used to verify that the tile structure did not form any other undesired structures (usually shown as upper bands above the single intact band) and any break down structures (lower band indicating unstable dissociations) when annealed at stoichiometric amounts. Figure 8 shows AFM images taken of the 2D crystals that were formed using the Tiamat design. The images prove that Tiamat was able to design the structure and then generate a sequence that could fold into the rationally designed array structures.

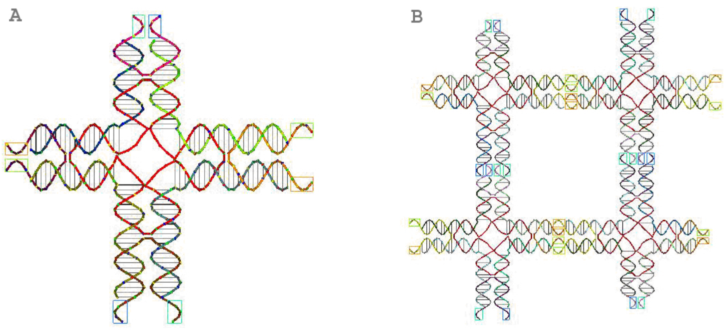

Figure 6: Tiamat model of the 4x4 DNA tile and a model of the joining of four DNA tiles together

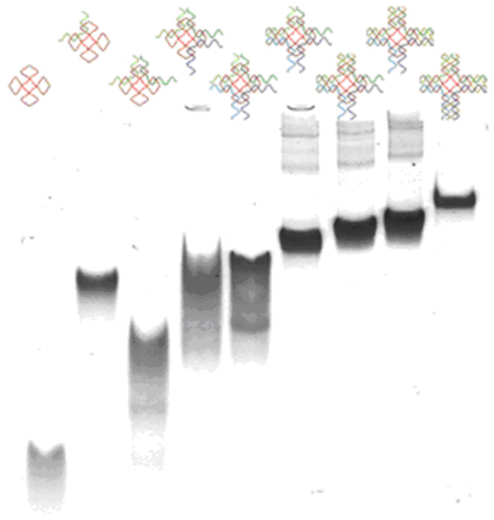

Figure 7: Native PAGE gel of the formation of the 4x4 DNA tile. Tiamat models are used to represent each lane. The complete tile is from a purified sample of the 4x4 tile

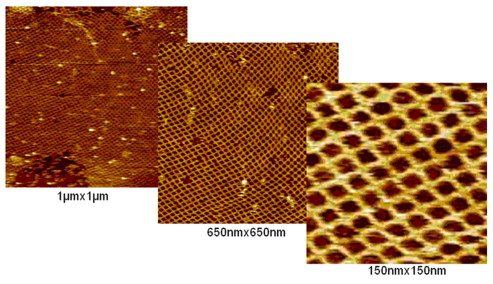

Figure 8: AFM images of the 2D crystals formed by the 4x4 DNA tile

4 Discussion

As demonstrated by the above examples and figures, we have shown that Tiamat is a novel and useful program for the designing of DNA structures and modeling those structures for feasibility. The main novelty of Tiamat is the possibility to efficiently model large and complex DNA structures. Future implementation and improvement of Tiamat may include more features of DNA modeling such as functions to estimate thermodynamic parameters for a designed DNA nanostructure.

Tiamat is written in C++ using Microsoft Foundation Classes (MFC) for windowing and interface, and OpenGL for rendering. Therefore, it only runs on Microsoft Windows operating systems; specifically, Windows 2000, XP, and Vista. Tiamat requires only OpenGL 1.0, so it can run on virtually any available video card.

Supplemental information includes a manual for the Tiamat software, detailed model and sequences of the 4x4 DNA tile and experimental methods used in this work. The Tiamat program and manual can be downloaded free from http://chemistry.asu.edu/faculty/hao_yan.asp.

4.1 Acknowledgements

This work is supported by a grant from NIH to S.L. and H.Y.; H.Y. also would like to thank funding supports from NSF, ONR and AFOSR. We thank members from Hao Yan's group for testing the use of the Tiamat program.

References

[1] Seeman, N.C. Nucleic Acid Junctions and Lattices. *J. Theo. Biol.* 1982, 99, 237-247.

[2] **(a)** Kallenbach, N.R., Ma, R.-I. and Seeman, N.C. An Immobile Nucleic Acid Junction Constructed from Oligonucleotides. *Nature* 1983, 305, 829-831. **(b)** Chen, J., and Seeman, N.C. The Synthesis from DNA of a Molecule with the Connectivity of a Cube. *Nature* 1991, 350, 631-633. **(c)** Zhang, Y., and Seeman, N.C. The Construction of a DNA Truncated Octahedron. *J. Am. Chem. Soc.* 1994, 116, 1661-1669. **(d)** Fu, T.-J. and Seeman, N.C. DNA Double Crossover Structures. *Biochemistry* 1993, 32, 3211-3220. **(e)** Shih W.M., Quispe J.D., and Joyce, G.F. A 1.7-kilobase single-stranded DNA that folds into a nanoscale octahedron. *Nature* 2004, 427, 618-621. **(f)** Goodman, R.P., Schaap, I.A.T., Tardin, C.F, Erben, C.M., Berry, R.M., Schmidt, C.F., and Turberfield, A.J. Rapid chiral assembly of rigid DNA building blocks for molecular nanofabrication. *Science* 2005, 310, 1661-1665. **(g)** Erben, C.M., Goodman, R.P., Turberfield, A.J. A Self-Assembled DNA Bipyramid. *J. Am. Chem. Soc.* 2007, 129, 6992-6993.

[3] **(a)** Winfree, E., Liu, F., Wenzler, L.A., and Seeman, N.C. Design and Self-Assembly of Two-Dimensional DNA Crystals. *Nature* 1998, 394, 539-544. **(b)** LaBean, T., Yan, H., Kopatsch, J., Liu, F., Winfree, E., Reif, J.H., and Seeman, N.C. The Construction of DNA Triple Crossover Molecules. *J. Am. Chem. Soc.* 2000, 122, 1848-1860. **(c)** Shen, Z., Yan, H., Wang, T., and Seeman, N.C. Paranemic Crossover DNA: A Generalized Holliday Structure with Applications in Nanotechnology. *J. Am. Chem. Soc.* 2004, 126, 1666-1674. **(d)** Mao, C., Sun, W., and Seeman, N. C. Designed Two-Dimensional DNA Holliday Junction Arrays Visualized by Atomic Force Microscopy. *J. Am. Chem. Soc.* 1999, 121, 5437-5443. **(e)** Yan, H., Park, S.H., Ginkelstein, G., Reif, J.H., and LaBean, T.H. DNA templated Self-assembly of Protein Arrays and Highly Conductive Nanowires. *Science* 2003, 301, 1882-1884. **(f)** Liu, D., Wang, M., Deng, Z., Walulu, R., and Mao, C. Tensegrity: Construction of Rigid DNA Triangles with Flexible Four-Arm DNA Junctions. *J. Am. Chem. Soc.* 2004, 126, 2324-2325. **(g)** He, Y., Chen, Y., Liu, H., Ribbe, A. E., and Mao, C. Self-Assembly of Hexagonal DNA Two-Dimensional (2D) Arrays. *J. Am. Chem. Soc.* 2005, 127, 12202-12203. **(h)** He, Y., Tian, Y., Ribbe, A. E., and Mao, C. Highly Connected Two-Dimensional Crystals of DNA Six-Point-Stars. *J. Am. Chem. Soc.* 2006, 128, 15978-15979. **(i)** Reishus, D., Shaw, B., Brun, Y., Chelyapov, N., and Adleman, L. Self-Assembly of DNA Double-Double Crossover Complexes into High-Density, Doubly Connected, Planar Structures. *J. Am. Chem. Soc.* 2005, 127, 17590-17591. **(j)** Ke, Y., Liu, Y., Zhang, J., and Yan, H. A Study of DNA Tube Formation Mechanisms Using 4-, 8-, and 12-Helix DNA Nanostructures. *J. Am. Chem. Soc.* 2006, 128, 4414-4421.

[4] Seeman, N.C. The Interactive Manipulation and Design of Macromolecular Architecture Utilizing Nucleic Acid Junctions. *J. Mol. Graphics* 1985, 3, 34-39.

[5] Seeman, N.C. Physical Models for Exploring DNA Topology. *J Biomol. Struct. Dynam.* 1988, 5, 997-1004.

[6] Birac, J.J., Sherman, W.B., Kopatsch, J., Constantinou, P.E. and Seeman, N.C. GIDEON, A Program for Design in Structural DNA Nanotechnology. *J. Mol. Graphics Model.* 2006, 25, 470-480.

[7] Seeman, N.C. De Novo Design of Sequences for Nucleic Acid Structure Engineering. *J. Biomol. Struct. Dynam.* 1990, 8, 573-581.

[8] Introduction to Algorithms. Cormen et al.

[9] Numerical Optimization, Springer Series in Operations Research and Financial Engineering. Jorge Nocedal, Stephen Wright.

[10] An Introduction to Optimization, Second Edition. Edwin K. P. Chong, Stanislaw H. Żak.

[11] Level of Detail for 3D Graphics. The Morgan Kaufmann Series in Computer Graphics. David Luebke, Martin Reddy, Jonathan D. Cohen, Amitabh Varshney, Benjamin Watson, Robert Huebner.

[12] Rothemund, P.W.K. Folding DNA to create nanoscale shapes and patterns. *Nature* 2006, 440, 297-302.