Published online DD MM YYYY

Nucleic Acids Research, YYYY, Vol. xx, No. xx ***1–6***
doi:10.1093/nar/gkn000

DNAforge: A design tool for nucleic acid wireframe nanostructures

Antti Elonen, [1,*] **Leon Wimbes,** [1,2] **Abdulmelik Mohammed** [3] **and Pekka Orponen** [1,*]

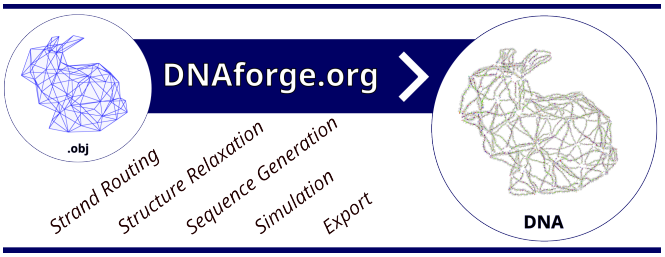
[1]Department of Computer Science, Aalto University, Finland [2]Department of Mathematics and Computer Science, Philipps University of Marburg, Germany [3]Department of Biomedical Engineering, San José State University, USA

Received 2024-02-17; Revised YYYY-MM-DD; Accepted YYYY-MM-DD

ABSTRACT

DNAforge is an online tool that provides a unified, user-friendly interface to several recent design methods for DNA and RNA wireframe nanostructures, with the possibility to integrate additional methods into the same framework. Currently, DNAforge supports three design methods for DNA nanostructures and two for RNA nanostructures. The tool enables the design, visualisation and sequence generation for highly complex wireframe nanostructures with a simple fully automated process. DNAforge is freely accessible at `https://dnaforge.org/`.

GRAPHICAL ABSTRACT



INTRODUCTION

In nucleic acid nanotechnology, nanoscale structures are self-assembled from rationally designed strands of DNA or RNA (1, 2). The base-pairing characteristics of nucleic acids make them a finely programmable fabrication material, which enables the assembly of structures with high precision and complexity, comprising currently up to tens of thousands of nucleotides.

DNA and RNA origami (3, 4) are two powerful, broadly applicable design paradigms that set forth how a long baseline strand ("scaffold strand" in DNA origami) can be guided to fold into a desired target shape or structure by a carefully programmed array of auxiliary strands or motifs ("staple strands" in DNA origami, kissing-loops and other connector motifs in RNA origami). Both methods have been used to design a large variety of 2D shapes and 3D structures (5, 6).

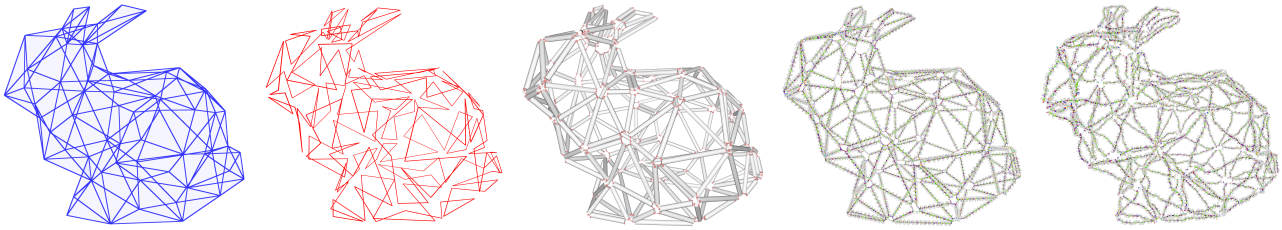
Most current 3D origami designs follow the approach of packing several 2D layers of helices or helix bundles on top of each other, and/or curving helix bundles as initially suggested in (7, 8). An alternative path to 3D design is to create a *wireframe* structure that incorporates only the boundary edges and vertices of the 3D model. There have been several notable pre-origami excursions in this direction (9, 10), but it has mostly started to gain a following with the development of the flexible and robust origami techniques (6, 11). Some advantages of wireframe designs as compared to helix-packing ones include economy of strand use, which allows the construction of larger structures, and better folding under low-salt conditions. Some of the challenges, on the other hand, are the sometimes low rigidity of the structures, particularly for large single-helix edge designs (this can be alleviated by employing multi-helix edges, at the cost of increased strand use) and the low yield of large and complex designs.

There already exist several nucleic acid nanostructure design tools (8, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21). Most of these however address helix-packing designs, with the more recent ones oriented towards wireframe structures including vHelix (14), DAEDALUS (15) and ATHENA (18) for 3D DNA wireframes, Sterna (20) for single-stranded 3D RNA wireframes and PyDAEDALUS (21) for 3D RNA/DNA hybrid wireframes. These tools however mostly support one specific design approach each and many are also off-line, requiring a separate process for installing the tool and its ancillary libraries, which may sometimes be difficult to locate or in the worst case deprecated.

THE DNAFORGE TOOL

DNAforge is a web application that provides a user-friendly unified interface and an extensible framework for the automated design of nucleic acid wireframe nanostructures. The tool currently covers five design methods with complementary characteristics, and is open to future extensions. It produces designs that include the full 3D nucleotide model, stapling arrangement where applicable, and

*To whom correspondence should be addressed. Email: antti.elonen@aalto.fi, pekka.orponen@aalto.fi



(a) Mesh model (b) Routing model (c) Cylinder model (d) Nucleotide model (e) Simulation

Figure 1. The DNAforge design workflow.

the primary sequences. The designs can also be exported to the widely-used oxView tool (22, 23) for further editing, visualisation and analysis, and to the oxDNA molecular dynamics engine (24, 25, 26) for simulation. DNAforge also provides an option for in-tool oxDNA simulation, if the engine and a connecting backend module are locally installed.

DNAforge is written in the TypeScript enhancement of JavaScript, converted into JavaScript ES6 and bundled with the Webpack JavaScript bundler for use in a browser. It makes significant use of the Three.js 3D rendering library and the Cannon-es 3D physics engine. The code's dependencies on these and other libraries are managed with the package manager NPM. The DNAforge user interface is built on the same Metro4 UI library as that of oxView, which should make it easily approachable to the nucleic acid nanotechnology community.

At present, DNAforge supports five design methods: the A-trail and spanning-tree based design methods for scaffolded DNA origami from (14) and (15), respectively; a cycle cover -based method for scaffold-free DNA wireframes that extends the one presented in (27) (extension unpublished), the spanning tree -based method for RNA origami from (20) and its enhancement (unpublished). These methods are briefly summarised below, and we refer the reader to the original publications for the detailed descriptions. The DNAforge workflows are nearly identical for all the methods, as outlined below.

DESIGN WORKFLOW

The first step in the DNAforge design workflow (Figure 1) is to create a 3D mesh model in the standard OBJ format (Figure 1a). This can be done using some 3D modelling suite such as Blender (28) or Maya (29), and a large variety of models are also available on the Internet. This input mesh is then used as the basis for one of the available design methods, each of which produces a routing model, a cylinder model, and a nucleotide model.

The abstract routing model is produced first. It is then converted to a cylinder model, where each cylinder represents a double helix. The strain in the system can optionally be relaxed via a physical simulation that tries to minimise the length of each helix-to-helix connection, while still preventing helical overlaps. The cylinder model is then used to generate a nucleotide-level model, which fully determines the 3D structure of the designed DNA or RNA nanostructure. Finally, a primary structure can be generated. The full 3D design can be exported as a UNF file (30) for further processing by other tools, and the strand sequences can be exported as a CSV file for ordering and synthesis.

Routing model

The routing model (Figure 1b) represents the paths one or more strands take around the wireframe of the mesh. For the A-trail method, the model consists of a scaffold path that follows a topologically constrained Eulerian circuit traversing around a reconditioned wireframe. For the Spanning-tree/DNA, Spanning-tree/RNA, and Xuong-tree/RNA methods, it represents a single strand path that traverses twice around the spanning tree of the wireframe. For the Cycle-cover method, it represents a number of directed cycles that cover the wireframe by traversing each edge once in both directions. The strand paths in the five routing methods currently implemented are sketched on a Schlegel diagram of a tetrahedron in Figure 4. The details of these methods are presented in the Design Methods section.

The routing model defines how the edges of the wireframe model should connect to each other, i.e., it determines the connections of the individual cylinders of the cylinder model.

Cylinder model

DNAforge adopts the common abstraction of a double helix as a cylinder. The cylinder model is used to calculate the exact number of nucleotides aligned along each wireframe edge and to avert physical overlaps between neighbouring helices. Each cylinder has a specific length and radius and contains four connection points representing the two 5' and two 3' ends of a double helix.

Cylinders are positioned and connected to each other according to the routing model. Their sizes are determined by a scale-parameter provided by the user. The scale parameter converts the length units of the input mesh into nanometers. At a scale of 1 nm, an edge of length 1 would be converted to a cylinder of length 1 nm. The positions of a cylinder's connection points and its diameter depend on the type of the cylinder model (DNA or RNA) and on the number of nucleotides aligned on it, which in turn depends on the length of the cylinder. For a DNA cylinder, the diameter is always fixed at 2 nm, corresponding to the mean diameter of a physiological B-type DNA double helix. For an RNA-cylinder, the diameter is 2.3 nm, corresponding to an A-type RNA double helix.

Technically, the scale, position, and orientation of a cylinder are represented by a 4×4 matrix. This allows for the

nucleotides generated based on the cylinder to be mapped to their global coordinates by simple matrix multiplication.

The cylinder model can optionally be relaxed using a physics simulation implemented on the cannon-es physics engine. The cylinders are modelled as cylinder-shaped rigid bodies, with their connection points joined together with constrained springs. The springs try to pull neighbouring cylinders together, but collisions between the rigid bodies prevent them from overlapping. The relaxation procedure tries to minimise the lengths of the linker strands between connected cylinders, which is useful when using a variable number of linker nucleotides in the design or when exporting the nucleotide model for simulation.

Nucleotide model

The nucleotide model (Figure 1d) is generated from the cylinder model. Each cylinder is converted into two antiparallel strands, and their 5' and 3' ends are connected to each other with short linker segments according to the connectivity of the cylinder model. Depending on the design method, the strands are also re-routed further: For instance, some cylinders in the ST-RNA design method should form kissing loops, which are created by changing the connectivity of certain nucleotides in the middle of the strands.

The individual nucleotides are generated and oriented by first creating a regular DNA or RNA double helix along the Y-axis. The number of nucleotides is determined by the length of the cylinder in nucleotides. The double helix is then rotated and positioned to match the target cylinder's rotation and position, i.e, the transformation matrix of each nucleotide is multiplied by the cylinder's transformation matrix. The linker strands between cylinders are generated by spherical linear interpolation between the positions of the 3' and 5' ends of the strands connected by the linker.

Once the linker strands are added, the nucleotide model consists of long cyclic strands. Strand gaps, or nicks, can be added to these to break the cycles into shorter splints ("staple strands" in scaffolded DNA origami), and make these as short as possible without compromising binding stability. The longer the overlap between two strands, the stronger they bind to each other. However, the length of these binding domains also affects the total length of the strands, which is what our nicking procedure tries to minimise. In DNAforge, the minimum overlap is a user-defined parameter.

After the nucleotide model is generated and the nucleotides that are to be base paired have been identified, appropriate complementary strand sequences can be devised. For the scaffold-based design methods, the scaffold strand sequence – commonly the M13mp18 viral genome or one of its variants – determines the sequences of the staple strands. DNAforge also provides the option for a user to upload a custom scaffold sequence. The ST-RNA method, on the other hand, utilises for a part of the sequence design the NUPACK tool (31), for which DNAforge provides an export function, and the Cycle-cover method has its own primary structure generator based on a local search algorithm.

The completed nucleotide model can then be exported as a UNF file for further visualisation, editing and simulation, and the strand sequences can be exported as a CSV file towards an eventual synthesis of the structure.

Simulation

DNAforge also provides the possibility of in-tool simulation and visualisation by the oxDNA molecular dynamics engine (24, 25, 26). The tool comes with an optional backend that acts as a wrapper around oxDNA, exposing many of its capabilities to DNAforge via a REST API and a WebSocket. The wrapper is written in Kotlin and can either be run using the provided Docker Compose file or by installing oxDNA locally and compiling the wrapper.

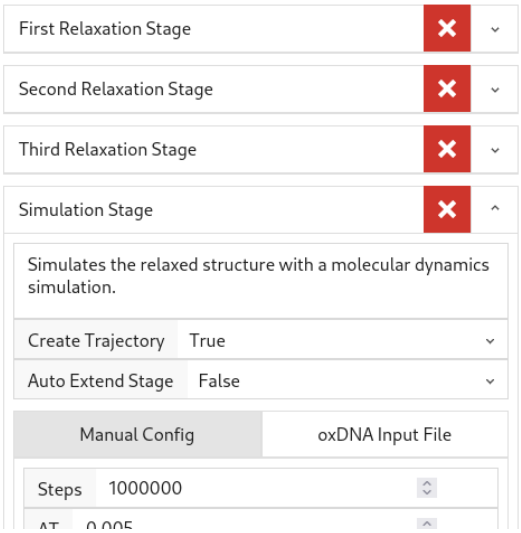


Figure 2. Configuring simulation jobs on DNAforge.

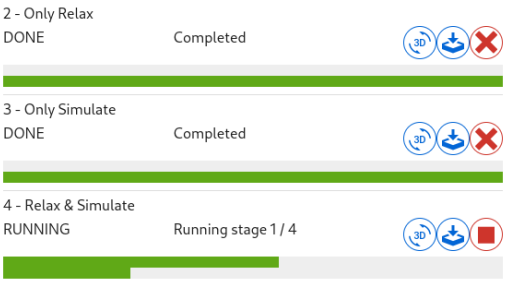
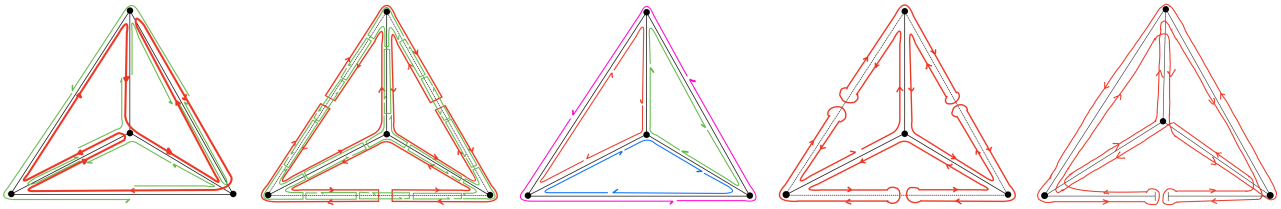


Figure 3. Running simulation jobs on DNAforge.

Using this backend, a given nucleotide model can be simulated and visualised live directly from the DNAforge interface, as demonstrated by the snapshot of a nucleotide model in simulation in Figure 1e.

OxDNA simulations invariably need a pre-simulation relaxation process to get the nucleotide model into an acceptable initial conformation, lest the simulation "blow up" because of nonphysically strong simulated interaction forces. This relaxation process can be quite complex, requiring several stages with different methods and durations.

By default, this relaxation process comprises three stages: a simple potential energy minimisation, a Monte Carlo simulation, and finally a molecular dynamics simulation with a very small integration time step. The more complex the structures are, the longer the individual relaxation stages need to run.



(a) AT-DNA (b) ST-DNA (c) CC-DNA (d) ST-RNA (e) XT-RNA

Figure 4. Schematics of design methods currently supported by DNAforge: A-trail/DNA (14), Spanning tree/DNA (15), Cycle-cover/DNA (27), Spanning-tree/RNA (20), Xuong-tree/RNA (unpublished).

Parameters for these default stages can be easily modified via the DNAforge web interface (Figure 2) before submitting a simulation job to the queue (Figure 3) at the backend.

Perhaps most importantly, the backend also supports adaptive extension of the relaxation stages, based on potential energy reductions observed in the process. Thus the user does not need to estimate the appropriate lengths of the relaxation stages, or find these by trial and error, but the backend intrinsically observes the progress of the relaxation and terminates only when the structure seems to have reached a stable conformation.

DESIGN METHODS

A-trail/DNA

In the A-trail or "BScOR/vHelix" approach to scaffolded DNA origami design (14), the scaffold strand (red curve in Figure 4a) is traced along the edges of a wireframe with minimal repetition, while simultaneously constraining the turns at every junction to be either a sharp left or a sharp right. For the notion of sharp-turns to be well-defined, all the edges of the wireframe must be incident to polygonal faces, which when glued together must form a 2-manifold/surface mesh.

The DNAforge search algorithm for A-trails employs a branch-and-bound algorithm over two possible configurations of sharp-turn choices at the junctions, leading to a worst-case exponential time complexity in the number of junctions. Nevertheless, the heuristics and pruning incorporated in the branch-and-bound algorithm almost always find A-trails for triangulated wireframe structures that can be assembled from the typical $\sim$7 kb scaffolds.

When the polygonal mesh is a topological sphere, the sharp turn condition of A-trail routings ensures that the designed scaffold route is unknotted (32), matching with the unknotted topological state of the standard circular scaffold strands in test tubes. While DNAforge can also search for and typically finds A-trails for toroidal meshes, it offers no guarantees about their unknottedness (32, 33). In addition to the automated A-trail search, DNAforge also supports manual import of A-trails, if for instance a user wants to use unknotted A-trails for toroidal meshes that were generated manually or using other tools.

Furthermore, selected single-helix A-trail edges can be reinforced into multihelix ones at the click of a button (34).

Spanning-tree/DNA

The Spanning-tree or "DAEDALUS" design method for DNA (15), presented in Figure 4b, routes the scaffold strand twice around a spanning tree of the input wireframe. (A spanning tree is a connected set of edges that covers every vertex.) In the ST-DNA implementation presented in (15), the spanning tree is found by Prim's algorithm, which produces a maximally branching tree. Each spanning-tree edge is rendered as a two-helix bundle that is held together by staple crossovers. Non-spanning-tree edges are similarly realised as two-helix bundles but have scaffold crossovers between the helices. The helix lengths are rounded to an integer number of turns to facilitate the specific stapling-pattern of the method. The staple sequences are set to be the Watson-Crick complements, per paired segment, to the scaffold strand sequence chosen by the user.

The linear-time ST-DNA method in DNAforge can find a routing for any reasonably-sized connected wireframe quickly. The routing is also guaranteed to be an unknot, although the geometry of the double helices at the vertices might cause the nucleotide model to be knotted. Since the ST-DNA method uses two double helices for each edge, the resulting structure can be expected to be more rigid than single double-helix designs. However, as each edge consumes twice as much of the scaffold strand, the largest structure that can be designed is half as big as that of the A-trail approach for Eulerian wireframes.

Cycle-cover/DNA

The Cycle-cover method, illustrated in Figure 4c, is a scaffold-free approach where a set of cyclic strands are arranged along a set of graph-theoretic cycles that cover each edge twice in antiparallel directions. The double covering cycles are built in a bottom up manner from strand crossovers at vertices, where it is ensured that the arms of the junctions are connected. The cyclic strands are then nicked in a staggered manner to yield the desired routings of the short linear strands. The lengths of the linear strands and the extent of their overlapping regions are controlled by user-defined parameters. The Cycle-cover method can generate scaffold-free designs for all connected wireframes.

The strand sequences in the Cycle-cover method are generated with the Focused Metropolis Search (35) algorithm. This local search algorithm tries to minimise the length of the longest repeated substring to avoid non-specific and unintended pairings while adhering to the user-supplied constraints on GC content, linker bases, and prohibited

subsequences. Fully complementary non-specific pairings can exist only if repeated substrings exist, but a repeated substring does not necessarily mean that there is potential for a non-specific pairing. The distinction, however, is not very restrictive, which is why it was chosen as an optimisation target. Due to the computational complexity, only substrings contained entirely within strands are considered, rather than subsequences or substrings spanning across more than one strand.

Spanning-tree/RNA

The Spanning-tree or "Sterna" method (20), presented in Figure 4d, is an RNA origami technique that routes a single linear RNA strand twice around a spanning-tree to form the A-helices of the core secondary structure. It then bulges out hairpin motifs towards the middle of the non-tree edges so that each non-tree edge is rendered as 180° kissing loops. The kissing loops behave much like regular double helices and as such are modelled with normal cylinders in the cylinder model. The open 5'-to-3' nick is placed at the longest wireframe edge.

The primary structure for an ST-RNA design can either be generated entirely randomly, subject to Watson-Crick complementarity conditions, or it can be generated externally and imported back into DNAforge. DNAforge has an export function which creates a NUPACK-runnable input file, where the kissing loops and certain specific bases are already set. The output of NUPACK can then be imported back into DNAforge. The kissing loop sequences are selected from a pre-generated list, which contains sequence pairs that form strong and highly specific bonds.

The DNAforge implementation of the ST-RNA method has similar limitations and guarantees as the ST-DNA method: It allows for the routing of any connected graph, and the route is guaranteed to be an unknot, but the nucleotide model can still be knotted due to the geometry of the double helices. ST-RNA can be used to generate highly complex structures with hundreds of kissing loops, but experimentally it seems that structures with more than a few dozen kissing loops have difficulties in folding correctly and/or tend to form large aggregates of nonspecifically paired particles, unless experimental conditions are very carefully controlled (20).

Xuong-tree/RNA

The Xuong-tree method (Figure 4e) is an RNA strand routing technique that minimises the number of kissing loops in the eventual structure. It achieves this by finding for the input mesh a specific spanning tree, a Xuong Tree, that minimises the number of odd-sized components in its co-tree, viz. its edge-complement graph (36, 37). A stably bound single-stranded route can be constructed around this spanning tree and the even-sized components of the co-tree, but one kissing loop is necessary for each of the odd-sized components. For so called upper-embeddable graphs, which includes all fully triangulated meshes (38) at most a single kissing loop in total is needed. As with Sterna, the strand gap in an XT-RNA design is placed at the longest mesh model edge. Due to the highly pseudoknotted nature of XT-RNA designs, the primary structure is currently generated randomly, subject to Watson-Crick pairing conditions.

The XT-RNA algorithm can be used on any connected graph, and the route can be resolved as an unknot on spherical meshes. DNAforge offers no guarantees about the unknottedness of routes on non-spherical meshes. Since structures generated with the XT-RNA method typically contain at most a single kissing loop, it can potentially be used to create more complex successfully folding and less aggregation-prone RNA structures than the ST-RNA method.

CONCLUSION

We have introduced DNAforge, a software tool for designing 3D wireframe nucleic acid nanostructures. The tool is hosted on GitHub (`https://github.com/dnaforge/`) under the MIT license and can be accessed at `https://dnaforge.org/`. This website is free and open to all users and there is no login requirement.

ACKNOWLEDGEMENTS

The DNAforge software development was initiated by a wish to modernise the vHelix design tool built at Björn Högberg's lab at Karolinska Institutet. We are very grateful to him and Erik Benson for their suggestions and feedback on several versions of the new tool. The name "DNAforge" was suggested by Björn.

Conflict of interest statement. None declared.

REFERENCES

1. Nadrian C. Seeman and Hanadi F. Sleiman. DNA nanotechnology. *Nature Reviews Materials*, 3:17068, November 2017. [doi:10.1038/natrevmats.2017.68].
2. Ofer I. Wilner, Doron Yesodi, and Yossi Weizmann. RNA nanostructures: from structure to function. *Bioconjugate Chemistry*, 34(1):30–36, January 2023. [doi:10.1021/acs.bioconjchem.2c00417].
3. Paul W. K. Rothemund. Folding DNA to create nanoscale shapes and patterns. *Nature*, 440(7082):297–302, March 2006. [PubMed:16541064] [doi:10.1038/nature04586].
4. Cody Geary, Paul W. K. Rothemund, and Ebbe S. Andersen. A single-stranded architecture for cotranscriptional folding of RNA nanostructures. *Science*, 345(6198):799, August 2014. [PubMed:25124436] [doi:10.1126/science.1253920].
5. Swarup Dey, Chunhai Fan, Kurt V. Gothelf, Jiang Li, Chenxiang Lin, Longfei Liu, Na Liu, Minke A. D. Nijenhuis, Barbara Saccà, Friedrich C. Simmel, Hao Yan, and Pengfei Zhan. DNA origami. *Nature Reviews Methods Primers*, 1(1):13, January 2021. [doi:10.1038/s43586-020-00009-8].
6. Erik Poppleton, Niklas Urbanek, Taniya Chakraborty, Alessandra Griffo, Luca Monari, and Kerstin Göpfrich. RNA origami: design, simulation and application. *RNA Biology*, 20(1):510–524, December 2023. [doi:10.1080/15476286.2023.2237719].
7. Shawn M. Douglas, Hendrik Dietz, Tim Liedl, Björn Högberg, Franziska Graf, and William M. Shih. Self-assembly of DNA into nanoscale three-dimensional shapes. *Nature*, 459(7245):414–418, May 2009. [PubMed:19458720] [PubMed Central:PMC2688462] [doi:10.1038/nature08016].
8. Hendrik Dietz, Shawn M. Douglas, and William M. Shih. Folding DNA into twisted and curved nanoscale shapes. *Science*, 325(5941):725, August 2009. [PubMed:19661424] [PubMed Central:PMC2737683] [doi:10.1126/science.1174251].
9. Junghuei Chen and Nadrian C. Seeman. Synthesis from DNA of a molecule with the connectivity of a cube. *Nature*, 350(6319):631–633, April 1991. [doi:10.1038/350631a0].
10. William M. Shih, Joel D. Quispe, and Gerald F. Joyce. A 1.7-kilobase single-stranded DNA that folds into a nanoscale octahedron. *Nature*, 427(6975):618–621, February 2004. [doi:10.1038/nature02307].

11. Pekka Orponen. Design methods for 3D wireframe DNA nanostructures. *Natural Computing*, 17(1):147–160, March 2018. [doi:10.1007/s11047-017-9647-9].
12. Shawn M. Douglas, Adam H. Marblestone, Surat Teerapittayanon, Alejandro Vazquez, George M. Church, and William M. Shih. Rapid prototyping of 3D DNA-origami shapes with caDNAno. *Nucleic Acids Research*, 37(15):5001–5006, August 2009. [doi:10.1093/nar/gkp436].
13. Sean Williams, Kyle Lund, Chenxiang Lin, Peter Wonka, Stuart Lindsay, and Hao Yan. Tiamat: a three-dimensional editing tool for complex DNA structures. In *DNA Computing: 14th International Meeting on DNA Computing*, volume 5347 of *Lecture Notes in Computer Science*, pages 90–101. Springer, 2009. [doi:10.1007/978-3-642-03076-5_8].
14. Erik Benson, Abdulmelik Mohammed, Johan Gardell, Sergej Masich, Eugen Czeizler, Pekka Orponen, and Björn Högberg. DNA rendering of polyhedral meshes at the nanoscale. *Nature*, 523(7561):441–444, July 2015. [PubMed:26201596] [doi:10.1038/nature14586].
15. Rémi Veneziano, Sakul Ratanalert, Kaiming Zhang, Fei Zhang, Hao Yan, Wah Chiu, and Mark Bathe. Designer nanoscale DNA assemblies programmed from the top down. *Science*, May 2016. [PubMed:27229143] [PubMed Central:PMC5111087] [doi:10.1126/science.aaf4388].
16. Elisa de Llano, Haichao Miao, Yasaman Ahmadi, Amanda J Wilson, Morgan Beeby, Ivan Viola, and Ivan Barisic. Adenita: interactive 3D modelling and visualization of DNA nanostructures. *Nucleic Acids Research*, 48(15):8269–8275, September 2020. [PubMed:32692355] [PubMed Central:PMC7470936] [doi:10.1093/nar/gkaa593].
17. David Doty, Benjamin L Lee, and Tristan Stérin. scadnano: A browser-based, scriptable tool for designing DNA nanostructures. In *26th International Conference on DNA Computing and Molecular Programming*, volume 174 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 9:1–9:17, Dagstuhl, Germany, 2020. Schloss Dagstuhl - Leibniz-Zentrum für Informatik. [doi:10.4230/LIPIcs.DNA.2020.9].
18. Hyungmin Jun, Xiao Wang, Molly F Parsons, William P Bricker, Torsten John, Shanshan Li, Steve Jackson, Wah Chiu, and Mark Bathe. Rapid prototyping of arbitrary 2D and 3D wireframe DNA origami. *Nucleic Acids Research*, 49(18):10265–10274, 2021. [PubMed:34508356] [PubMed Central:PMC8501967] [doi:10.1093/nar/gkab762].
19. Nicolas Levy and Nicolas Schabanel. ENSnano: A 3D modeling software for DNA nanostructures. In *27th International Conference on DNA Computing and Molecular Programming*, volume 205 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 5:1–5:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. [doi:10.4230/LIPICS.DNA.27.5].
20. Antti Elonen, Ashwin Karthick Natarajan, Ibuki Kawamata, Lukas Oesinghaus, Abdulmelik Mohammed, Jani Seitsonen, Yuki Suzuki, Friedrich C. Simmel, Anton Kuzyk, and Pekka Orponen. Algorithmic design of 3D wireframe RNA polyhedra. *ACS Nano*, 16:16608–18816, September 2022. [PubMed:36178116] [PubMed Central:PMC9620399] [doi:10.1021/acsnano.2c06035].
21. Molly F. Parsons, Matthew F. Allan, Shanshan Li, Tyson R. Shepherd, Sakul Ratanalert, Kaiming Zhang, Krista M. Pullen, Wah Chiu, Silvi Rouskin, and Mark Bathe. 3D RNA-scaffolded wireframe origami. *Nature Communications*, 14(1):382, January 2023. [doi:10.1038/s41467-023-36156-1].
22. Erik Poppleton, Joakim Bohlin, Michael Matthies, Shuchi Sharma, Fei Zhang, and Petr Šulc. Design, optimization and analysis of large DNA and RNA nanostructures through interactive visualization, editing and molecular simulation. *Nucleic Acids Research*, 48(12):e72–e72, July 2020. [PubMed:32449920] [PubMed Central:PMC7337935] [doi:10.1093/nar/gkaa417].
23. Joakim Bohlin, Michael Matthies, Erik Poppleton, Jonah Procyk, Aatmik Mallya, Hao Yan, and Petr Šulc. Design and simulation of DNA, RNA and hybrid protein–nucleic acid nanostructures with oxView. *Nature Protocols*, 17(8):1762–1788, August 2022. [PubMed:35668321] [doi:10.1038/s41596-022-00688-5].
24. Petr Šulc, Flavio Romano, Thomas E. Ouldridge, Lorenzo Rovigatti, Jonathan P. K. Doye, and Ard A. Louis. Sequence-dependent thermodynamics of a coarse-grained DNA model. *The Journal of Chemical Physics*, 137(13):135101, 2012. [PubMed:23039613] [doi:10.1063/1.4754132].
25. Lorenzo Rovigatti, Petr Šulc, István Z. Reguly, and Flavio Romano. A comparison between parallelization approaches in molecular dynamics simulations on GPUs. *Journal of Computational Chemistry*, 36(1):1–8, 2015. [PubMed:25355527] [doi:10.1002/jcc.23763].
26. Petr Šulc, Flavio Romano, Thomas E. Ouldridge, Jonathan P. K. Doye, and Ard A. Louis. A nucleotide-level coarse-grained model of RNA. *The Journal of Chemical Physics*, 140(23), 2014. [PubMed:24952569] [doi:10.1063/1.4881424].
27. Wen Wang, Silian Chen, Byoungkwon An, Kai Huang, Tanxi Bai, Mengyuan Xu, Gaëtan Bellot, Yonggang Ke, Ye Xiang, and Bryan Wei. Complex wireframe DNA nanostructures from simple building blocks. *Nature Communications*, 10(1):1067, March 2019. [PubMed:30842408] [PubMed Central:PMC6403373] [doi:10.1038/s41467-019-08647-7].
28. The Blender Foundation. Blender: The free and open source 3D creation suite. `https://blender.org/`. [Online; accessed 16-February-2024].
29. Inc. Autodesk. Autodesk Maya 3D animation and visual effects software. `https://autodesk.com/maya/`. [Online; accessed 16-February-2024].
30. David Kuťák, Erik Poppleton, Haichao Miao, Petr Šulc, and Ivan Barišić. Unified Nanotechnology Format: One way to store them all. *Molecules*, 27(1), 2022. [PubMed:35011301] [PubMed Central:PMC8746876] [doi:10.3390/molecules27010063].
31. Joseph N. Zadeh, Conrad D. Steenberg, Justin S. Bois, Brian R. Wolfe, Marshall B. Pierce, Asif R. Khan, Robert M. Dirks, and Niles A. Pierce. NUPACK: Analysis and design of nucleic acid systems. *Journal of Computational Chemistry*, 32(1):170–173, January 2011. [PubMed:20645303] [doi:10.1002/jcc.21596].
32. Abdulmelik Mohammed, Nataša Jonoska, and Masahico Saito. The topology of scaffold routings on non-spherical mesh wireframes. In *26th International Conference on DNA Computing and Molecular Programming*, volume 174 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 1:1–1:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. [doi:10.4230/LIPIcs.DNA.2020.1].
33. Ada Morse, William Adkisson, Jessica Greene, David Perry, Brenna Smith, Jo Ellis-Monaghan, and Greta Pangborn. DNA origami and unknotted A-trails in torus graphs. *Journal of Knot Theory and Its Ramifications*, 29(07):2050041, 2020. [doi:10.1142/S0218216520500418].
34. Marco Lolaico, Sebbe Blokhuizen, Boxuan Shen, Yang Wang, and Björn Högberg. Computer-aided design of A-trail routed wireframe DNA nanostructures with square lattice edges. *ACS Nano*, 17(7):6565–6574, April 2023. [PubMed:36951760] [PubMed Central:PMC10100577] [doi:10.1021/acsnano.2c11982].
35. Sakari Seitz, Mikko Alava, and Pekka Orponen. Focused local search for random 3-satisfiability. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(06):P06006, June 2005. [doi:10.1088/1742-5468/2005/06/P06006].
36. Merrick L Furst, Jonathan L Gross, and Lyle A McGeoch. Finding a maximum-genus graph imbedding. *Journal of the ACM (JACM)*, 35(3):523–534, 1988. [doi:10.1145/44483.44485].
37. Jernej Rus. Antiparallel d-stable traces and a stronger version of Ore problem. *Journal of Mathematical Biology*, 75:109–127, 2017. [PubMed:27853820] [doi:10.1007/s00285-016-1077-2].
38. Ladislav Nebeskỳ. Every connected, locally connected graph is upper embeddable. *Journal of Graph Theory*, 5(2):205–207, 1981. [doi:10.1002/jgt.3190050211].