A general design method for scaffold-free DNA wireframe nanostructures

Antti Elonen[1], Abdulmelik Mohammed[2], and Pekka Orponen[1]

[1] Department of Computer Science, Aalto University, Finland
antti.elonen@aalto.fi, pekka.orponen@aalto.fi
[2] Department of Biomedical Engineering, San José State University, USA
abdulmelik.mohammed@sjsu.edu

Abstract. In the area of DNA nanotechnology, approaches to composing wireframe nanostructures exclusively from short oligonucleotides, without a coordinating long scaffold strand, have been proposed by Goodman et al. (2004) and Wang et al. (2019). We present a general design method that extends these special cases to arbitrary wireframes, in the sense of graphs linearly embedded in 2D or 3D space. The method works in linear time in the size of the given wireframe model and is already available for use in the online design tool *DNAforge*. We also interpret the method in terms of topological graph embeddings, which opens up further research opportunities in developing this design approach.

Keywords: DNA origami, DNA wireframes, scaffold-free nanostructure design, strong anti-parallel traces, cycle covers, topological graph embeddings

1 Introduction

The research area of DNA nanotechnology [26] uses DNA as a generic, highly programmable building material for creating nanoscale structures and devices. Since almost 20 years now, the technique of *DNA origami*, introduced by Paul Rothemund in 2006 [24] has been the leading approach to DNA nanostructure design. In DNA origami, a long natural *scaffold strand*, typically the ca. 7200 nt (nucleotides) long cyclic genome of the M13 bacteriophage, is guided to fold into the desired shape by a large number of ca. 20–200 nt long *staple strands*. Over the years, this technique has been used to create an amazing collection of nanostructures for a wide variety of interests and purposes [2].

A recent direction of interest have been *wireframe* 2D and 3D DNA nanostructures, where very general design methods based on the origami approach have already been developed [1,29,23], together with fully automated tools [1,29,16,8]. Sparse wireframe structures have the prospective advantage of being more strand-efficient and more stable in natural low-salt conditions than the commonly used helix-packed designs [4,3].

A critical limitation of the otherwise extremely versatile and robust DNA origami design method is its dependence on the global scaffold strand. For ambitious designs, the ∼7200 nt scaffold length provided by the M13 genome variants

is quite limited, and while research into longer strands has provided some remarkable demonstrations (e.g. [21,35]), these novel strands are not widely available, and because of little experience with them may also carry unidentified weaknesses such as decreased product yield.

As an alternative route forward, methods for *scaffold-free* nanostructure designs have been proposed [34,33,18,22,32]. These designs comprise only short single-stranded oligonucleotides, which are interleaved to constitute the target structure. Using the scaffold-free approach, designs 100 times larger than what is achievable using a single M13 scaffold strand have been synthesised [22], although the product yield starts to decrease for really large designs.

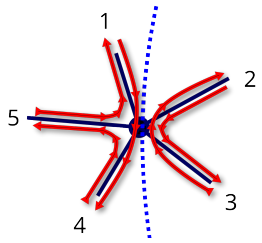
For similar reasons as in scaffolded DNA origami, there is emerging interest in *wireframe* scaffold-free nanostructures. The earliest precedent of these kinds of designs is possibly the (pre-origami) DNA tetrahedron by Goodman et al. [12], where the wireframe structure was constituted by routing the four faces of a tetrahedron by one 55-nt oligonucleotide each, in counterclockwise direction. The base sequences in these single-stranded oligos were designed so that for each edge of the tetrahedron, the antiparallel segments of the oligos on both sides of the edge were perfectly matching, in the complementary Watson-Crick pairing sense. (And also so that there were not too long nonspecific pairing domains elsewhere.) This idea was generalised and further developed by Wang et al. [32], who designed a number of convex 3D polyhedral wireframes and 2D wireframe lattices using this approach, and also some 3D cubic lattices by a different approach based on combining 6-arm vertex motifs.

We continue this line of research by presenting in Section 2 a simple and efficient design method that works for *any* reasonable 2D or 3D wireframe model, that is, a graph linearly embedded in space. This method is already implemented and available for use in our online design tool *DNAforge* (`https://dnaforge.org`) [8]. Section 3 then discusses the task of finding good nucleotide sequences for DNA wireframes created using this design method, and Section 4 introduces an interesting connection to topological graph embeddings. Section 5 explains the use of the *DNAforge* tool with design examples, and Section 6 presents some concluding remarks and suggestions for further work.

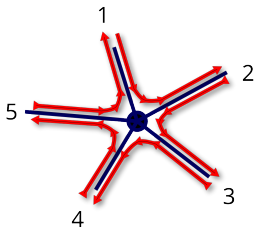
2 The Cycle-Cover Design Method

Our general goal is to render a given wireframe model, which we take to be a finite, connected simple graph, with no leaf vertices, that is linearly embedded in 2D or 3D space, as a DNA structure with single-duplex edges. This entails two conditions on the DNA strand arrangement: (i) each edge of the target wirefame must be rendered by two strand segments that traverse the edge in anti-parallel directions (ignoring for the moment any inter-oligo nicks that may be located on the edge), and (ii) the crossover arrangement of the strands meeting at each vertex v must be such that v is *stable*. By this we mean that if one considers two edges e and e' incident to v as *locally connected at* v when there is a strand segment that enters v along e and exits along e' or vice versa, then these local

edge connections in the neighbourhood of v form a simple cycle. (The local routing pattern of the strands at a vertex is called a "transition" in [10,6] and the local edge-connectivity graph the "vertex figure" in [9].)



(a) An unstable vertex that has two disjoint repetition neighbourhoods.



(b) A stable vertex whose neighbours constitute a single repetition neighbourhood.

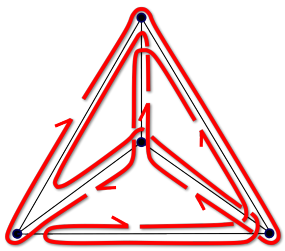
Fig. 1

An alternative view of condition (ii) is provided by the concept of *repetitions* at a vertex v (Figure 1) [9]. A *repetition neighbourhood* of a vertex v is a set N of neighbouring vertices, such that every strand segment that enters v from N also exits v to N. A vertex is then stable, if and only if all its neighbours constitute a single repetition neighbourhood. Figure 1 illustrates how an unstable vertex is at risk of becoming detached in a DNA rendering.

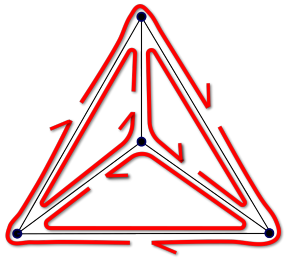
The conditions (i) and (ii) were recognised by Fijavž et al. [9] in their notion of a *strong anti-parallel trace*, by which they mean a closed edge walk on the graph that traverses each edge twice, in opposite directions, and so that no vertex has disjoint repetition neighbourhoods. (A complementary perspective on this approach is provided in [5], where the vertex stability condition is called "cyclic compatibility".)

Strong anti-parallel traces give a good characterisation of feasible single-strand designs for DNA wireframes. They, however, exist only in limited cases: more precisely if and only if every component of any co-tree of the target wireframe graph is of even size [9]. (Co-trees are the edge-complements of spanning trees in a graph [14, p. 751].) Many interesting polyhedra lack this characteristic. For instance, a tetrahedron has six edges and four vertices, so any spanning tree of a tetrahedron has three edges and so does its co-tree; which means that every co-tree must have at least one odd-sized component. (In fact all co-trees of a tetrahedron are connected and of size three.) This means that a tetrahedron does not admit a strong anti-parallel trace, and consequently does not have a good single-stranded rendering in DNA: any anti-parallel double trace of a tetrahedron contains at least one unstable vertex (Figure 2(a)).

However, in a scaffold-free setting one does not need to be constrained to a single trace for covering the wireframe. Both of the conditions (i) and (ii) can



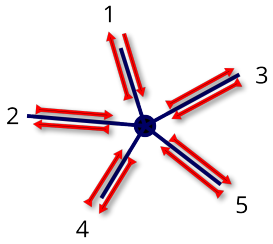
(a) An anti-parallel double trace of a tetrahedron, with an unstable vertex.



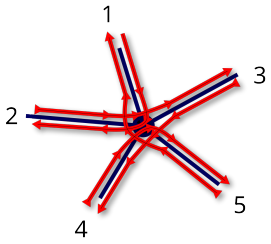
(b) An s.a.p. cycle cover of a tetrahedron.

Fig. 2

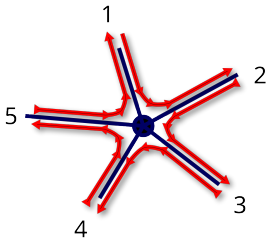
also be satisfied by covering the graph with multiple oriented edge cycles. This idea gives rise to the concept of a *strong antiparallel cycle cover* (briefly s.a.p. cycle cover). An s.a.p. cycle cover likewise traverses each edge twice, once in both directions, in a way that satisfies the vertex stability requirement, but it consists of multiple cyclic traces. (We are somewhat abusing the term "cycle" here, because our covering cycles may repeat edges, and so are precisely speaking closed walks on the graph.) We note that while the use of antiparallel cycle double covers for the construction of abstract DNA graphs was suggested already by Ellis-Monaghan in [6], the vertex stability condition was addressed there only in terms of antiparallel single-cycle covers that have no repetition neighbourhoods of size 1, which is not sufficient in the case of vertices of degree greater than three.



(a) A vertex with incident edges split into pairs of antiparallel directed edges.



(b) A routing around the vertex in numerical order: $1 \to 2 \to 3 \to 4 \to 5 \to 1$



(c) A visually more pleasing representation of the routing in (b).

Fig. 3

One can aim to construct s.a.p. cycle covers in two simple ways: either top-down or bottom-up. One first ensures vertex stability by assigning to each vertex v of degree d some local edge-traversal order by numbering the edges incident to v cyclically as $(1, 2, \ldots, d)$. Then, whenever a cycle enters vertex v along edge i, it must exit along edge $(i \bmod d) + 1$. As will be discussed in Section 4, this procedure is equivalent to finding facial cycles of an embedding of the graph in some orientable surface, and results in a single repetition neighbourhood at each vertex, i.e., the routing will be strong. The construction is illustrated in Figure 3.

If the input mesh constitutes an orientable surface, there exists a natural clockwise traversal order on it that has many nice properties, but in principle any family of cyclic permutations of the edges around the vertices induces an embedding of the mesh wireframe in *some* orientable surface (see Section 4).

Not all edge permutations may result in physically favourable routings, however, and some care should be taken if the input mesh does not constitute a surface. In that case, a reasonable edge permutation around a vertex v can be determined by finding a shortest Hamiltonian cycle in a metric connectivity graph around v. This graph is constructed by adding one new vertex on each of the edges incident to v at one unit distance away from v, and by fully connecting all of these new vertices to each other. This procedure minimises the total length of the eventual spacer nucleotide segments between strands, and should also mitigate nucleotide congestion at the vertices. (A refined version of this approach in the setting of scaffold-strand routings is proposed in [7].)

In the top-down approach one then initiates the construction of the first cyclic walk in the cover from some vertex v_0 with degree $d_0 = \deg(v_0)$ along the local edge 1, say. Then one follows the vertices' local routing rules, until one eventually returns to vertex v_0 along edge d_0. (The walk on the graph may of course have revisited the vertex v_0 along other edges already earlier.) After closing this first walk, one checks if there still remains some edge $\{u, v\}$ that has not yet been double-covered. If so, then one chooses either vertex u or v as the new start vertex, depending on which of them has a free exit direction (both will have if the edge $\{u, v\}$ has not yet been traversed in either direction), and initiates a new cyclic walk from there.

The bottom-up approach is even simpler. One first splits each undirected edge $\{u, v\}$ in the wireframe graph in two directed edges (u, v) and (v, u). Suppose then that edge $\{u, v\}$ has index i in the local numbering at u, and index j in the local numbering at v. Then we give the edges (u, v) and (v, u) local identities as $(u, v) = \mathrm{out}(u, i) = \mathrm{in}(v, j)$ and $(v, u) = \mathrm{out}(v, j) = \mathrm{in}(u, i)$. Now the s.a.p. cover cycle ensemble can be constructed bottom-up, without any touring of the graph, by including in it for every vertex u, of degree d_u, the walk segments $[\mathrm{in}(u, i), \mathrm{out}(u, i + 1)]$, for $i = 1, \ldots, d_u \pmod{d_u}$. (Experimentally, a similar approach as this was already used implicitly by Wang et al. in their design of $4\times4\times4$ and $8\times8\times4$ cubical lattices in [32].)

With proper data structures and bookkeeping, both approaches can be implemented to run in linear time, that is in time $O(|E|)$, where $|E|$ is the number of edges in the wireframe graph. For instance, the top-down method is essentially

Hierholzer's classical (1873) linear-time algorithm for finding an Eulerian cycle in an Eulerian graph [17, Algorithm 2.3.1], adapted to Eulerian digraphs and simplified by not merging the intermediate cycles into a single covering cycle.

3 Sequence Design

When the cycle cover is converted into DNA strands, it will initially consist of a number of long cyclical strands, which are then cut into shorter individual strands without compromising the structural stability of the wireframe. The longer the hybridised area between two overlapping antiparallel strands is, the stronger their chemical bond. However, this length also affects the total length of the strand, which should ideally be as short as possible, since long strands are difficult and expensive to produce.

Given a minimum hybridisation length parameter, N, determining the precise placement of strand gaps, or nicks, becomes a global optimisation problem. Short edges, in particular, present a challenge, as they lack sufficient space for accommodating multiple strand gaps. This means that adding a strand gap to one strand at one particular edge might prevent adding a strand gap to another strand at all.

For instance, an edge shorter than $2N$ nucleotides lacks sufficient space for even a single strand gap, while an edge of length $2N$ can accommodate only one strand gap on only one of the two strands. On the other hand, an edge spanning $5N$ nucleotides provides enough room for at least two strand gaps on each strand.

Once the strand gaps are placed, some strands will traverse at least a portion of two different edges and a possible single-stranded segment of linker/spacer nucleotides in between. These strands tend to be the longest, and minimising their lengths poses the trickiest problem. An edge that spans $5N$ nucleotides allows for these vertex-adjacent strands to be as short as possible on both sides of the edge, facilitating the nicking process for long edges. Short edges, however, remain a challenge, since placing a strand gap at one strand might impede nicking its complementary strand.

Our simple cycle cover nicking algorithm places strand gaps greedily for each edge of length $3N$ or more, i.e., edges that admit one or more nicks at both strands, so that the strands neighbouring the vertices are as short as possible. Next, the algorithm selects all edges of lengths from $2N$ to $3N - 1$ and selects the longer of the two strands and places the strand gap on that one. If any cyclical strands remain, or if any strand is longer than the maximum allowed, the nicking procedure fails and the algorithm needs to be run again with different parameters.

The primary sequence for a cycle cover design is generated with a local search algorithm previously described in [8]. The search algorithm, based on the Focused Metropolis Search method [27], aims to minimise the length of the longest repeated substring to avoid non-specific and unintended pairings, while also adhering to the given constraints of GC-content, forced bases, and prohib-

ited subsequences. Although fully complementary non-specific pairings can exist only if repeated substrings exist, a repeated substring does not necessarily mean that there is potential for a non-specific pairing. Since this distinction is not very restrictive, it was chosen as the optimisation target. Due to computational complexity, the algorithm only considers substrings contained entirely within strands, rather than subsequences or substrings spanning across more than one strand.

The search algorithm works by first assigning each nucleotide a random complementary base and then checking if the constraints are satisfied. If not, a random base corresponding to an unsatisfied constraint is chosen, and it is randomly changed to another. If the number of unsatisfied constraints is decreased, the change is accepted. Otherwise it is reverted with a probability of $1 - \eta$. This process is repeated until all constraints are satisfied. Then, the algorithm uses the same procedure with the repeated substrings: If a repeat of length R exists, a random base associated with such a repeat is changed, and the change is again accepted if no constraints were broken and if the number of repeats decreased. Otherwise, it is reverted with a probability of $1 - \eta$. This search process is iterated for K cycles, or until no repeats of length R exist. Afterwards, if repeats do exist, R is increased by one, and otherwise it is decreased by one. The search algorithm can then be run again with the updated value of R.

This algorithm generally performs well. A strand of length L even without any secondary structure whatsoever must contain repeated substrings of length $\lfloor (log_4 L) \rfloor$, and this algorithm typically yields primary sequences with the longest repeat equal to that number or larger by only one unless the constraints are unusually restrictive.

4 Cycle Covers and Graph Embeddings

The s.a.p. cycle cover design problem can be formulated in terms of graph embeddings in surfaces. We recall that a *(closed) surface* is a compact topological space where every point has a neighbourhood topologically equivalent to the plane and that a surface is *orientable* if it does not contain the Möbius strip. The classification theorem of closed orientable surfaces states that such a surface is either a sphere or a sphere with g handles attached, where g is the genus of the surface [20]. For instance, the surface of the bunny mesh model 4, which is the union of all the points in all the polygons (including their interiors), is topologically equivalent to a sphere.

A *graph embedding* in a surface is an injective mapping of the graph vertices to surface points, along with a mapping of the edges to arcs in the surface, such that a) the interiors of the arcs are disjoint, b) for each edge, the ends of its arc meet the points associated with the end vertices of the edge and c) no vertex is mapped to the interior of an arc. Removing the image of the graph from the surface results in a collection of distinct regions called *faces*, and when these faces are all topological open disks the embedding is said to be a *2-cell embedding*.

In a mesh model such as the bunny in Figure 4, the surface is given in 3D as the boundary of the solid, and the embedded graph comprises the points and lines of the mesh. However, graphs can also be embedded in abstract representations of surfaces, such as that of the torus as a square where the horizontal and vertical boundaries are appropriately identified to be glued. For a target wireframe like the $3\times3\times3$ lattice in Figure 4, no polygonal faces are present, and an embedding of the abstract graph in an abstract surface will be needed for interpreting the s.a.p double covers in terms of graph embeddings in surfaces.

An s.a.p cycle cover of a wireframe is equivalent to a double cover by face-bounding cycles of a 2-cell embedding of the abstract graph in some orientable closed surface. The equivalence of s.a.p cycle covers and facial cycle covers is evident by looking at *pure rotation systems* that capture the combinatorial structure of 2-cell graph embeddings in orientable surfaces [13]. A *pure rotation system* $\Pi(G)$ of a graph G assigns to each vertex v in the graph a cyclic permutation π_v of the edges incident to v. A 2-cell embedding of a graph induces a pure rotation system by fixing the π_v to be the clockwise order of the arcs in the local neighborhood of v. Conversely, a pure rotation system $\Pi(G)$ uniquely determines, up to orientation preserving equivalence, a graph embedding in an orientable surface. The faces of the embedding can be identified using a face tracing algorithm that generates the oriented bounding cycles of the faces [13, p. 115]. For each vertex incident edge pair (v, e), the rotation system induces an *oriented transition* $(e, v, \pi_v(e))$. The face tracing algorithm finds the oriented boundary cycles of the faces by joining the transitions. For example, the two transitions $(e, v, \pi_v(e))$, $(\pi_v(e), w, \pi_w(\pi_v(e))$ can be naturally joined as $(\ldots, e, v, \pi_v(e), w, \pi_w(\pi_v(e), \ldots)$.

There are many ways of selecting a pure rotation system for an abstract graph. If the target wireframe is the edge-skeleton of a given underlying orientable surface mesh, the s.a.p. cycle cover method chooses the pure rotation system that is induced by clockwise cyclic orderings of edges incident to the vertices. If the target wireframe is given without an underlying surface mesh, the s.a.p. cycle cover method extracts a pure rotation system from the geometric proximity of local edges in the real 3-space of the wireframe, as described in Section 2.

The set of cycles generated by the method can be seen as the oriented boundary cycles $\mathcal{C} = \{C_1, \ldots, C_k\}$ of the faces by the face tracing algorithm. There are four oriented transitions associated with an edge $e = \{u, v\}$ in the equivalent rotation system: the two transitions $(e, u, \pi_u(e))$, $(\pi_u^{-1}(e), u, e)$ at u and the two transitions $(\pi_v^{-1}(e), v, e)$, $(e, v, \pi_v(e))$ at v. Thus, the cycles in $\mathcal{C}$ that trace e will contain the segments $(\pi_u^{-1}(e), u, e, v, \pi_v(e))$ and $(\pi_v^{-1}(e), v, e, u, \pi_u(e))$, traversing e in antiparallel directions. One can easily show that in this case the repetition neighborhood N of any vertex v induced by $\mathcal{C}$ is the regular neighborhood of v, i.e. the set of all vertices adjacent to v: Let $\pi_v = (e_0, e_1, \ldots, e_{d-1})$ and $e_i = \{v, u_i\}$, for $0 \leq i \leq d-1$, where d is the degree of v. For any $0 \leq i \leq d-1$, the vertices u_i, $u_{i+1 \bmod d}$ are in the same repetition neighborhood because of the transition $(e_i, v, e_{i+1 \bmod d})$. Thus all u_i are in the same repetition neighborhood as desired for a stable vertex.

A graph can be cellularly embedded in orientable closed surfaces of different genera. The number of faces in the embedding is related to the genus through the generalised Euler's polyhedron formula $|V| - |E| + |F| = 2 - 2g$, where g is the genus of the surface and $|F|$ denotes the number of faces. The number of circular strands used by the s.a.p. cycle cover method is thus directly connected to the surface we choose to embed the graph into as the number of faces in the embedding. Along this line, Jonoska and Saito [15] have used thickened graph models to show that the maximum number of circular strands needed to assemble an abstract connected 3-regular graph is at most $\beta_1 + 1$, where $\beta_1 = |E| - |V| + 1$ is the co-tree size of the graph, also known as its first Betti number.

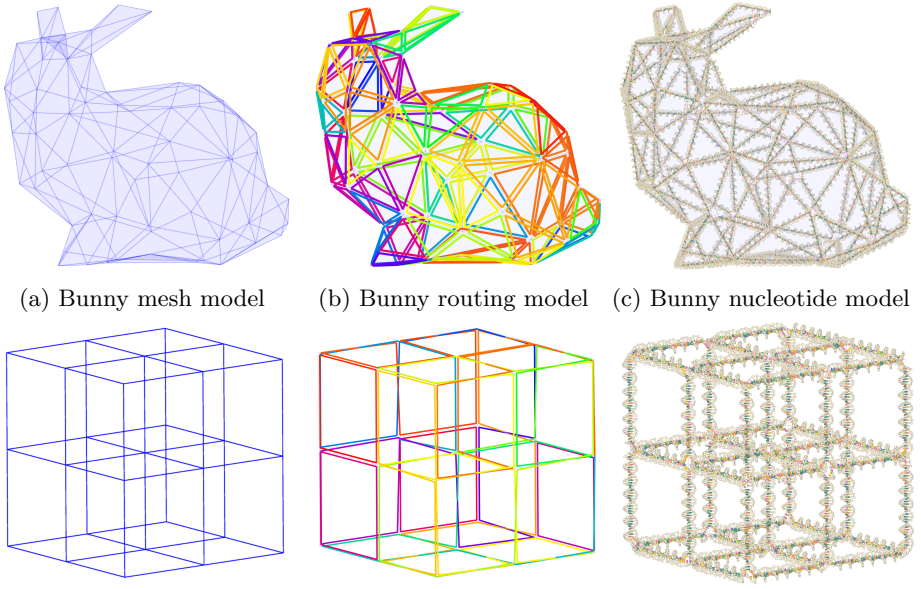
If the fewest possible circular strands is desirable, a maximum genus embedding should be used. Ideally, a one face embedding would result in a strong antiparallel cover by a single cycle, but as noted earlier this is only possible if and only if each connected component of the co-tree has an even number of edges [9]. Ellis-Monaghan [6] has proved for instance that the minimum number of circular strands needed to assemble the tetrahedral graph is two. Interestingly, the minimum number of circular strands needed for an s.a.p. cycle cover can actually be computed in polynomial time using maximum genus embeddings [11]. If, on the other hand, the number of circular strands should be maximised, a minimum genus embedding can be used. In general, deciding whether a graph can be cellularly embedded into a surface of a given genus g is NP-complete [28] and thus there will not likely be a polynomial time algorithm for determining the maximum number of circular strands for an s.a.p. cover.

5 Design Examples

The s.a.p. cycle cover method is integrated in the *DNAforge* web application, which provides an automated platform for designing nucleic acid nanostructures based on 3D wireframe models. With this method, labelled CC-DNA in the *DNAforge* user interface, users can generate s.a.p. cycle cover routings and nucleotide level models from any target wireframe with a single click. The workflow of the tool is illustrated in Figure 4, which showcases the examples of a bunny and a $3 \times 3 \times 3$ lattice. Note that while the bunny mesh model is provided with explicit faces and as embedded on a sphere-equivalent surface, the $3 \times 3 \times 3$ lattice mesh model contains just the wireframe, because the $3 \times 3 \times 3$ lattice as a 3D linearly embedded graph does not have any natural face structure.

The implementation offers users the flexibility to influence the design by adjusting several parameters such as the scale, minimum hybridisation length, and GC-content of the structures. The user can also optionally reduce strain in the designed structure with a DNA-duplex level physical simulation.

Once the DNA wireframe is designed, it can be exported in various formats, including as a PDB file, a UNF file [19] or as *oxDNA* files [31] for simulation or further editing, or as a CSV file containing the primary sequences. The nucleotide model can also be simulated directly from the interface, provided the *DNAforge*



(a) Bunny mesh model (b) Bunny routing model (c) Bunny nucleotide model

(d) 3×3×3 mesh model (e) 3×3×3 routing model (f) 3×3×3 nucleotide model

Fig. 4: The CC-DNA design workflow in *DNAforge* for a bunny and a $3\times3\times3$ lattice. The user first uploads a 3D model in the standard OBJ format (a, d). With a click of a button, *DNAforge* constructs an s.a.p. cycle cover routing for the mesh (b, e). The cycle cover is then automatically converted into a nucleotide-level model (c, f).

backend together with the *oxDNA* molecular simulation package [31,25,30] are locally installed.

6 Conclusions and Further Work

We have introduced a general and efficient design method for scaffold-free DNA wireframe nanostructures via strong antiparallel cycle covers. An implementation of the method is available on the online design tool DNAforge, accessible at `https://dnaforge.org/`.

In some application settings, the user might be interested in the minimum genus embedding of their wireframe model, and this can in principle be determined via the connection of strand count to embedding surface genus. The task is NP-complete, but a combinatorial search might work in many practical scenarios. Also exploring maximum-genus embeddings with the goal of minimising the strand count is of interest, and we are looking forward to experimental validations of the design method.

References

1. Benson, E., Mohammed, A., Gardell, J., Masich, S., Czeizler, E., Orponen, P., Högberg, B.: DNA rendering of polyhedral meshes at the nanoscale. Nature 523(7561), 441–444 (2015), doi: 10.1038/nature14586
2. Dey, S., Fan, C., Gothelf, K.V., Li, J., Lin, C., Liu, L., Liu, N., Nijenhuis, M.A.D., Saccà, B., Simmel, F.C., Yan, H., Zhan, P.: DNA origami. Nature Reviews Methods Primers 1(1), 13 (2021), doi: 10.1038/s43586-020-00009-8
3. Dietz, H., Douglas, S.M., Shih, W.M.: Folding DNA into twisted and curved nanoscale shapes. Science 325(5941), 725 (2009), doi:10.1126/science.1174251
4. Douglas, S.M., Dietz, H., Liedl, T., Högberg, B., Graf, F., Shih, W.M.: Self-assembly of DNA into nanoscale three-dimensional shapes. Nature 459(7245), 414–418 (2009), doi:10.1038/nature08016
5. Ellingham, M.N., Ellis-Monaghan, J.A.: Bi-eulerian embeddings of graphs and digraphs (2024), preprint arXiv:2404.00325, doi:10.48550/arXiv.2404.00325
6. Ellis-Monaghan, J.A.: Transition polynomials, double covers, and biomolecular computing. Congressus Numerantium 166, 181 (2004)
7. Ellis-Monaghan, J.A., McDowell, A., Moffatt, I., Pangborn, G.: DNA origami and the complexity of Eulerian circuits with turning costs. Natural Computing 14(3), 491–503 (2015), `https://doi.org/10.1007/s11047-014-9457-2`, doi:10.1007/s11047-014-9457-2
8. Elonen, A., Wimbes, L., Mohammed, A., Orponen, P.: DNAforge: A design tool for nucleic acid wireframe nanostructures. Nucleic Acids Research (to appear), preprint: https://research.cs.aalto.fi/nc/papers/dnaforge_2024.pdf
9. Fijavž, G., Pisanski, T., Rus, J.: Strong traces model of self-assembly polypeptide structures. MATCH Communications in Mathematical and in Computer Chemistry 71, 199–212 (2014), preprint: https://doi.org/10.48550/arXiv.1308.4024
10. Fleischner, H.: Eulerian Graphs and Related Topics. Part 1, Volume 1, Annals of Discrete Mathematics, vol. 45. North-Holland Publishing Co., Amsterdam (1990)
11. Furst, M.L., Gross, J.L., McGeoch, L.A.: Finding a maximum-genus graph imbedding. Journal of the ACM (JACM) 35(3), 523–534 (1988), doi: 10.1145/44483.44485
12. Goodman, R.P., Berry, R.M., Turberfield, A.J.: The single-step synthesis of a DNA tetrahedron. Chemical Communications 40(12), 1372–1373 (2004), doi: 10.1039/B402293A
13. Gross, J.L., Tucker, T.W.: Topological Graph Theory. Courier Corporation (2001)
14. Gross, J.L., Yellen, J., Zhang, P.: Handbook of Graph Theory, 2nd Ed. CRC Press (2014)
15. Jonoska, N., Saito, M.: Boundary components of thickened graphs. In: Jonoska, N., Seeman, N.C. (eds.) DNA Computing. pp. 70–81. Springer Berlin Heidelberg (2002), doi:10.1007/3-540-48017-X_7
16. Jun, H., Wang, X., Parsons, M.F., Bricker, W.P., John, T., Li, S., Jackson, S., Chiu, W., Bathe, M.: Rapid prototyping of arbitrary 2D and 3D wireframe DNA origami. Nucleic Acids Research 49(18), 10265–10274 (2021), doi: 10.1093/nar/gkab762
17. Jungnickel, D.: Graphs, Networks and Algorithms. Algorithms and Computation in Mathematics, Springer Berlin Heidelberg (2013)
18. Ke, Y., Ong, L.L., Shih, W.M., Yin, P.: Three-dimensional structures self-assembled from DNA bricks. Science 338(6111), 1177 (2012), doi: 10.1126/science.1227268
19. Kuťák, D., Poppleton, E., Miao, H., Šulc, P., Barišić, I.: Unified Nanotechnology Format: One way to store them all. Molecules 27(1) (2022), doi:10.3390/molecules27010063

20. Lee, J.: Introduction to Topological Manifolds, vol. 202. Springer Science & Business Media (2010)
21. Marchi, A.N., Saaem, I., Vogen, B.N., Brown, S., LaBean, T.H.: Toward larger DNA origami. Nano Letters 14(10), 5740–5747 (2014), doi: 10.1021/nl502626s
22. Ong, L.L., Hanikel, N., Yaghi, O.K., Grun, C., Strauss, M.T., Bron, P., Lai-Kee-Him, J., Schueder, F., Wang, B., Wang, P., Kishi, J.Y., Myhrvold, C., Zhu, A., Jungmann, R., Bellot, G., Ke, Y., Yin, P.: Programmable self-assembly of three-dimensional nanostructures from 10,000 unique components. Nature 552(7683), 72–77 (2017), doi: 10.1038/nature24648
23. Orponen, P.: Design methods for 3D wireframe DNA nanostructures. Natural Computing 17(1), 147–160 (2018), doi:10.1007/s11047-017-9647-9
24. Rothemund, P.W.K.: Folding DNA to create nanoscale shapes and patterns. Nature 440(7082), 297–302 (2006), doi: 10.1038/nature04586
25. Rovigatti, L., Šulc, P., Reguly, I.Z., Romano, F.: A comparison between parallelization approaches in molecular dynamics simulations on GPUs. Journal of Computational Chemistry 36(1), 1–8 (2015), doi:10.1002/jcc.23763
26. Seeman, N.C., Sleiman, H.F.: DNA nanotechnology. Nature Reviews Materials 3, 17068 (2017), doi:10.1038/natrevmats.2017.68
27. Seitz, S., Alava, M., Orponen, P.: Focused local search for random 3-satisfiability. Journal of Statistical Mechanics: Theory and Experiment 2005(06), P06006 (2005), doi:10.1088/1742-5468/2005/06/P06006
28. Thomassen, C.: The graph genus problem is NP-complete. Journal of Algorithms 10(4), 568–576 (1989), doi:10.1016/0196-6774(89)90006-0
29. Veneziano, R., Ratanalert, S., Zhang, K., Zhang, F., Yan, H., Chiu, W., Bathe, M.: Designer nanoscale DNA assemblies programmed from the top down. Science (2016), doi:10.1126/science.aaf4388
30. Šulc, P., Romano, F., Ouldridge, T.E., Doye, J.P.K., Louis, A.A.: A nucleotide-level coarse-grained model of RNA. The Journal of Chemical Physics 140(23), 235102 (2014), doi:10.1063/1.4881424
31. Šulc, P., Romano, F., Ouldridge, T.E., Rovigatti, L., Doye, J.P.K., Louis, A.A.: Sequence-dependent thermodynamics of a coarse-grained DNA model. The Journal of Chemical Physics 137(13), 135101 (2012), doi:10.1063/1.4754132
32. Wang, W., Chen, S., An, B., Huang, K., Bai, T., Xu, M., Bellot, G., Ke, Y., Xiang, Y., Wei, B.: Complex wireframe DNA nanostructures from simple building blocks. Nature Communications 10(1), 1067 (2019), doi:10.1038/s41467-019-08647-7
33. Wei, B., Dai, M., Yin, P.: Complex shapes self-assembled from single-stranded DNA tiles. Nature 485(7400), 623–626 (2012), doi: 10.1038/nature11075
34. Yin, P., Hariadi, R.F., Sahu, S., Choi, H.M.T., Park, S.H., LaBean, T.H., Reif, J.H.: Programming DNA tube circumferences. Science 321(5890), 824–826 (2008), doi:10.1126/science.1157312
35. Zhang, H., Chao, J., Pan, D., Liu, H., Huang, Q., Fan, C.: Folding super-sized DNA origami with scaffold strands from long-range PCR. Chemical Communications 48(51), 6405–6407 (2012), doi:10.1039/c2cc32204h