Unknotted Strand Routings of Triangulated Meshes

Abdulmelik Mohammed[1⋆] and Mustafa Hajij[2]

[1] Aalto University, Espoo, Finland,
abdulmelik.mohammed@aalto.fi
[2] University of South Florida, Tampa, Florida,
mhajij@usf.edu

Abstract. In molecular self-assembly such as DNA origami, a circular strand's topological routing determines the feasibility of a design to assemble to a target. In this regard, the Chinese-postman DNA scaffold routings of Benson et al. (2015) only ensure the unknottedness of the scaffold strand for triangulated topological spheres. In this paper, we present a cubic-time $\frac{5}{3}-$approximation algorithm to compute unknotted Chinese-postman scaffold routings on triangulated orientable surfaces of higher genus. Our algorithm guarantees every edge is routed at most twice, hence permitting low-packed designs suitable for physiological conditions.

Keywords: DNA origami · knot theory · graph theory · Chinese postman problem

1 Introduction

Since the pioneering work of Ned Seeman in 1982 [29], DNA has emerged as a versatile, programmable construction material at the nanoscale. Accordingly, DNA-based polyhedra [3,4,13,16,30,33], periodic- [36] and algorithmic [28] crystals, custom two- [2,27,34] and three-dimensional shapes [6,20] have since been demonstrated in the lab. With the introduction of the experimentally robust DNA origami technique [27], highly automated and well-abstracted derivative design methods [3,33] have become prominent, enabling design and synthesis of evermore complex three-dimensional geometries.

In the Chinese-postman-tour DNA origami design of triangulated topological spheres by Benson et al. [3], the long circular *scaffold* strand is first routed on the mesh skeleton and then held in place with hundreds of short *staple* strands. Henceforth, double helices, comprised half-and-half from the scaffold and staples, constitute the edges of the mesh, while the set of nearby strand transitions between edges form the vertices. However, the limitation of the method to topological spheres excludes simple but natural wireframes such as the nested cube synthesized by Veneziano et al. [33]. Evident with the view of the nested cube as a toroidal mesh, the class of higher-genus surfaces permits a much larger class of spatially embedded wireframes to be designed.

A fundamental topological constraint when employing a circular strand for assembly is that the strand routing must be unknotted. In a recent paper, Ellis-Monaghan et al. [8] have shown the scaffold routings of Benson et al. [3] can be knotted on higher-genus surfaces. In Figure 1, we present another example of a knotted Chinese-postman-tour/Eulerian-tour routing on a triangulated torus. We leave it to the reader to verify

⋆ Corresponding author.

that the routing corresponds to a trefoil knot on the torus. For higher-genus surfaces such as tori, a routing can also be knotted or unknotted depending on how the surface is embedded in real space. For instance, the Chinese-postman-tour/Eulerian-tour routing in Figure 2 is knotted if the embedding of the torus is knotted (as in Figure 3), but is unknotted if the embedding is standard. This can be verified by noting that the routing helically follows the meridional (horizontal) direction.

In this work, we examine the problem of finding unknotted Chinese-postman tours on the 1-skeleton of higher-genus triangulated surfaces. We present a cubic-time approximation algorithm to compute unknotted Chinese postman tours on such surfaces. Our algorithm further guarantees that the tour routes each edge at most twice, thus allowing low-packed helix bundle designs suitable for low-salt solutions [3,33].

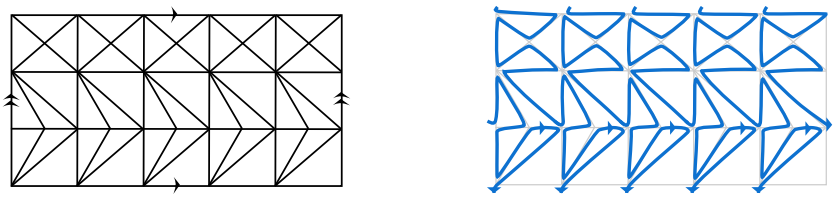

Fig. 1. Left: a planar representation of the 1-skeleton of a torus mesh. The torus is reconstructed, in a standard way, by glueing the horizontal boundaries together and likewise glueing the vertical boundaries together. Right: a knotted routing as a detached Chinese postman tour/Eulerian tour on the mesh skeleton. Note that since the boundaries of the planar representation are identified for glueing, only one copy of a boundary edge in the representation is visited by the Eulerian routing.

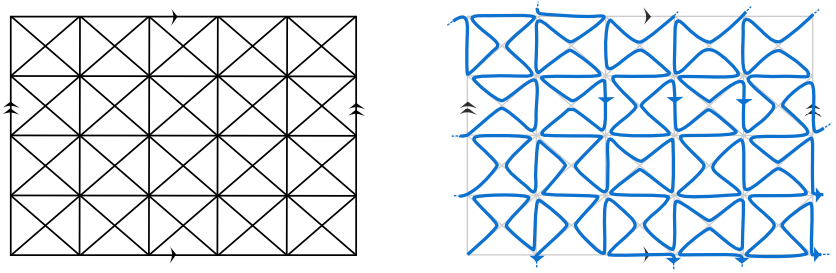

Fig. 2. Left: a planar representation of the 1-skeleton of a torus mesh. Right: a detached Chinese postman tour/Eulerian tour on the skeleton which is unknotted in a standard embedding of the torus, but is knotted in a knotted embedding of the torus. Note that a column of the mesh is a repeatable unit and thus this construction naturally leads to an infinite family of knotted and unknotted routings depending on the embedding of the torus in $\mathbb{R}^3$.

2 Preliminaries and Problem Definition

Assuming familiarity with basic topology [23] and graph theory [19], we only provide definitions to formally state our computational problem and prove our claims.

2.1 Triangulated Surfaces

A *surface* is a topological space that is Hausdorff, second countable, and locally homeomorphic to $\mathbb{R}^2$. We exclusively consider compact, connected, orientable surfaces, and for brevity refer to them as surfaces. Unless stated otherwise, a surface is assumed to be without boundary. The classification theorem of (compact, connected, orientable) surfaces states that any surface is homeomorphic to either the sphere or the connected sum of g tori, for $g \geq 1$. Surfaces without boundary can be classified up to homeomorphism via the *Euler characteristic* : $\chi(S) = 2 - 2g$, where g is the genus. Intuitively, the genus of a surface is the number of handles in that surface. For instance, a torus has genus one. A surface of genus zero and no boundary components is a *topological sphere*. A topological sphere with a single boundary component is called a *topological disk*. In practice surfaces are usually approximated by triangulated meshes (c.f. Figure 3), or more formally by simplicial complexes.

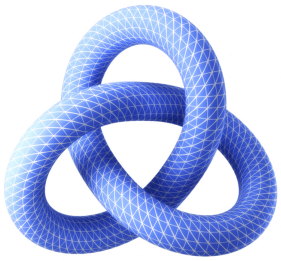

Fig. 3. A triangulated surface (torus) embedded in $\mathbb{R}^3$ in a knotted manner.

Let k be a positive integer and $A = \{v_0, \cdots, v_k\}$ be a set of points in $\mathbb{R}^n$. We say that $v_0, \cdots, v_k$ are *affinely independent*, if $v_0 - v_1, \cdots, v_0 - v_k$ are linearly independent.[3] A *k-dimensional simplex*, or simply a *k-simplex*, is the set $\sigma^k = [v_0, \cdots, v_k] = \{\sum_{i=0}^{k} \lambda_i v_i \in \mathbb{R}^n : \sum_{i=0}^{k} \lambda_i = 1, \lambda \geq 0\}$. Note that a simplex is completely determined by its set of vertices. We often call a 0-simplex a *vertex*, a 1-simplex an *edge* and a 2-simplex a *face*. Given a simplex σ_A on a set A, any non-empty subset T of A is also affinely independent and determines a simplex σ_T called a *facet* of σ_A.

A *k-dimensional simplical complex* is a finite collection Σ of simplices of dimension at most k that satisfies the following conditions. First, if σ is in Σ, then all the facets of σ are also in Σ. Second, if $\sigma_1, \sigma_2 \in \Sigma$, $\sigma_1 \cap \sigma_2 \neq \emptyset$, then $\sigma_1 \cap \sigma_2$ is in Σ. Last, every

[3] By convention, if $k = 0$, then v_0 is affinely independent.

point in Σ has a neighborhood that intersects at most finitely many simplices of Σ. The *i-skeleton* of a simplicial complex Σ is the union of the simplices of Σ of dimensions less than or equal to i. If Σ is a simplicial complex and $\sigma \in \Sigma$ then $star(\sigma) := \{\mu \in \Sigma \text{ such that } \mu \text{ contains } \sigma, \text{ or is a facet of a simplex which contains } \sigma\}$. We denote by $|\Sigma|$ the set obtained by taking the union of all simplices of a simplicial complex Σ, and equipped with the relative subspace topology of the usual topology of $\mathbb{R}^n$.

In this paper we only need 2-dimensional simplicial complexes. If Σ is a 2-dimensional simplicial complex then we denote by $V(\Sigma)$, $E(\Sigma)$ and $F(\Sigma)$ the set of vertices, edges and triangles of Σ, respectively. A *simplicial surface* S is a simplicial complex consisting of a finite set of faces such that (1) Every vertex in Σ belongs to at least one face in $F(\Sigma)$ and (2) For every v in $V(\Sigma)$, $|star(v)|$ is homeomorphic to a 2-disk. If Σ is a simplicial surface and in addition there is a piecewise-linear embedding $\mathcal{F} : |\Sigma| \longrightarrow \mathbb{R}^3$ then we call $(\Sigma, \mathcal{F})$ a *triangulated surface*, or simply a *mesh*. Note that if Σ is a triangulated surface then $|\Sigma|$ is a topological surface embedded in $\mathbb{R}^3$. Figure 3 shows an example of a triangulated surface. The Euler characteristic of a mesh Σ is given by $\chi(\Sigma) = |V(\Sigma)| - |E(\Sigma)| + |F(\Sigma)|$.

2.2 Postman Tours and Knots

In this paper, we only consider finite, undirected, loopless graphs. By graph, we mean a simple graph, reserving the term multigraph for graphs which can have parallel edges. We denote a multigraph as $G = (V, E)$, and use $V(G)$ and $E(G)$ to refer to its vertices and edges, respectively. A *walk* W of *length* $l \geq 0$ on a multigraph $G = (V, E)$ is a sequence of vertices and edges $(v_0, e_0, v_1, e_1, \cdots, e_{l-1}, v_l)$ such that $e_i = [v_{i-1}, v_i] \in E(G)$. We say W *visits* or *traces* an edge e if $e \in W$. A walk is said to be closed if $v_0 = v_l$. A *path* is a walk without repeated vertices and a *cycle* is like a path except that $v_0 = v_l$. A *trail* is a walk with distinct edges, and a trail is said to be *closed* if the walk is closed. A multigraph is said to be *connected* if there is a walk between any two vertices. A connected cycle-free graph is called a *tree* and more generally a cycle-free graph is called a *forest*.

A *postman tour* [7] is a closed walk which traces every edge at least once. The *length of a postman tour* is the length of the walk. A *Chinese postman tour* is a postman tour with minimum length. An *Eulerian tour* is a closed walk which visits every edge exactly once. A multigraph which admits an Eulerian tour is said to be *Eulerian*. A classical theorem of Euler [10] and Hierholzer [17] states that a multigraph is Eulerian if and only if it is connected and does not contain any odd-degree vertices. For an Eulerian multigraph, the notion of a Chinese postman tour and an Eulerian tour coincide (see the planar representations of Eulerian torus meshes and related routings in Figures 1 and 2.) In our work, we only work with postman tours on (simple) graphs. Moreover, we mostly view a postman tour on a graph as being an Eulerian tour on a related multigraph. In particular, the multiple traces of an edge by the postman tour are viewed as visits of independent copies of the edge in the Eulerian multigraph [7]. For further illustrations related to the concepts here, check the right most graph in Figure 6, a Chinese postman tour of this graph depicted on the right in Figure 7 and the related Eulerian multigraph shown on the left in Figure 7.

An embedding of a multigraph G in $\mathbb{R}^3$ is a representation of G in $\mathbb{R}^3$ where the vertices of G are represented by points on $\mathbb{R}^3$ and the edges of G are represented by simple arcs on $\mathbb{R}^3$ such that: (1) no two arcs intersect at interior points to either of them, (2) the two vertices defining an edge e are associated with the endpoints of the arc associated with e, and (3) there is no arc which includes points that are associated with other vertices. Here we are interested in multigraphs that are embedded on meshes. In particular, we are interested in the 1-*skeleton* of a surface mesh M which corresponds to the embedded graph $G = (V, E)$, where $V = V(M)$ and $E = E(M)$.

A *knot* in $\mathbb{R}^3$ is a (piece-wise) linear embedding of the circle S^1 in $\mathbb{R}^3$. Knots are considered up to ambient isotopy, that is two knots are said to be ambient isotopic if we can continuously deform one to the other without tearing or self intersection. An *unknot* is a knot ambient isotopic to the standard circle on the plane. Equivalently, an *unknot* is a knot that bounds an embedded piece-wise linear disk in $\mathbb{R}^3$ [24]. For a formal treatment of knot theory see [24,26]. Note that the standard definition of a postman tour as presented above is purely combinatorial and does not yet specify a curve to be analyzed as a knot. In the next section, we introduce an interpretation of a postman tour as a knot which permits any postman tour as a candidate solution for the unknotted Chinese postman tour routing problem (c.f. Problem 1.)

2.3 The Unknotted Chinese Postman Tour Problem

For a graph G embedded in $\mathbb{R}^3$, suppose the embedding of a postman tour T on G is the curve implied by the image of the vertex-edge sequence of the tour except that the repeated edges are mapped to parallel but non-overlapping curves which only meet at the endpoints. Consider the related Eulerian multigraph G'. Unless G' is a cycle, T repeats vertices, and hence the embedding of T either touches or intersects itself at vertices. For formal treatment of T as a knot, we need to make the embedding a simple curve while keeping the curve within the vicinity of G. For this purpose, we define the notion of a detachment of a postman tour embedding, which is simply a local unpinning of all the edge transitions at the vertices, as follows.

Let $T = (v_0, e_0, v_1, e_1, \cdots, e_{l-1}, v_l = v_0)$ be a postman tour on a graph G embedded in $\mathbb{R}^3$. Construct a new graph Δ with vertex set $\{\delta_0, \delta_1, \cdots, \delta_{l-1}\}$ and edge set $\{\alpha_0, \alpha_1, \cdots, \alpha_{l-1}\}$, where $\alpha_i = [\delta_i, \delta_{i+1}]$, for $0 \leq i \leq l-2$, and $\alpha_{l-1} = [\delta_{l-1}, \delta_0]$. Clearly, Δ is a cycle and has an Eulerian tour $T_\Delta = (\delta_0, \alpha_0, \delta_1, \alpha_1, \cdots, \alpha_{l-1}, \delta_l = \delta_0)$. Observe that while T might repeat vertices, T_Δ does not. Suppose we embed Δ as follows: (1) Each δ_i is at most a small $\epsilon > 0$ distance, in $\mathbb{R}^3$, away from v_i, (2) Each edge α_i is embedded exactly like e_i, except at its ends where it is incident to δ_i and δ_{i+1} instead of v_i and v_{i+1}. Note that by construction, all the vertices of Δ are at distinct locations. We call the induced embedding on T_Δ a *detachment* of T.[4] A detachment of a postman tour is simple and is thus a knot. We say that a *detached postman tour is unknotted* if the detachment is an unknot. Recall Figure 1 for an example of a knotted detachment of a Chinese postman tour on the 1-skeleton of a torus standardly embedded in $\mathbb{R}^3$. We now state our problem as follows.

[4] Detachments are also defined in the literature [11] for multigraphs without an associated embedding.

Problem 1. **Unknotted Chinese postman tour problem (UCPT)**: Given a triangulated oriented surface without a boundary and of genus $g \geq 1$, find a minimum length postman tour along its 1-skeleton which is detachable to an unknot.

An example input instance of **UCPT**, a genus one triangulated mesh, is shown on the left in Figure 4. Next, we present a $\frac{5}{3}$-approximation algorithm for **UCPT**, that is a postman tour with length at most two-thirds greater than any unknotted Chinese postman tour, which moreover guarantees that no edge is traced more than twice. Although the notion of detachment is important for the consideration of all possible postman tours as solutions to **UCPT**, our algorithm outputs non-crossing postman tours (c.f. Section 3.3 and Figure 7) where the detachments are clear without explicit construction.

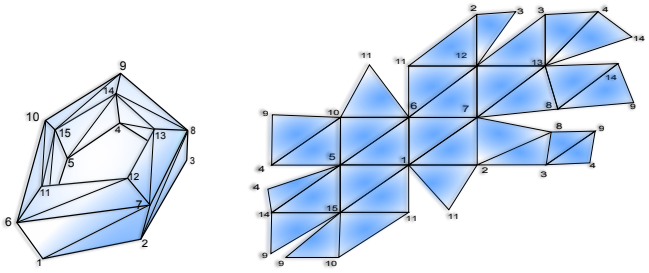

Fig. 4. Left: A toroidal mesh. Right: A polygonal schema of the mesh.

3 A Cubic Time Algorithm for Finding Unknotted Postman Tours

The main idea of our algorithm is to transfer the problem of finding an unknotted Chinese postman tour on an arbitrary surface mesh to the case of finding such a tour on the mesh's cutting to a topological disk. After such cutting, we can compute an unknotted Chinese postman tour on the disk, which then simply lifts to an unknotted approximate Chinese postman tour on the surface mesh. To be presented in detail in the subsequent sections, our algorithm, which we name as the **cut-and-route** algorithm, proceeds along the following steps:

1. Cut the input surface mesh M to obtain a topological disk D where a re-gluing of partnered boundary edges reconstructs M.
2. Remove one instance of the partnered edges of D and extract an embedded subgraph H on D whose edges are in a one-to-one correspondence with M.
3. Find a non-crossing Chinese postman tour on H, which then maps to the desired unknotted approximate Chinese postman tour on M.

3.1 Cutting a Surface to a Disk

Surface cutting is a problem that has been extensively studied due to its importance in surface parameterization and texture mapping [12,9]. For our algorithm, we adopt the

Algorithm 1: Mesh2disk: Cutting a triangulated surface to a disk.

Input: A triangulated surface M given as a face-to-face adjacency list.
Output: A polygonal schema D of M.

```
1  s ← the first face, D ← an empty adjacency list, Q ← empty queue of faces;
2  Enqueue s to Q;
3  for all faces f ∈ M do Mark f as not visited;
4  while Q is not empty do
5      f ← dequeue from Q;
6      Mark f as visited;
7      for g neighbor of f in M do
8          if g is not visited then
9              Enqueue g to Q;
10             Append g to D[f] and f to D[g];
11 return D;
```

basic algorithm of Dey and Schipper [5] for computing a polygonal schema. A *polygonal schema* [9] of a triangulated surface M is a topological disk that consists of all faces of M. A polygonal schema can be obtained by cutting a graph, called a *cut graph*, on the the 1-skeleton of M. Dey and Schipper's cutting algorithm starts with topological disk D which consists of a single face f on the surface M and keeps expanding D by gluing faces to its boundary. We present it in Algorithm 1, named as Mesh2disk, for analysis within our cut-and-route algorithm.

Mesh2disk is simply a breadth first search (BFS) on the dual graph implied by the face-to-face adjacency list of M, and the BFS tree represents the connectivity of the faces in D. For the torus mesh in Figure 4, the first few rounds of face addition are shown in Figure 5 and the resulting polygonal schema is shown on the right in Figure 4. Next, we prove three lemmas useful for the analysis of the cut-and-route algorithm.

Lemma 1. *Mesh2disk outputs a topological disk D.*

Proof. Let $(f_1, f_2, f_3, \cdots, f_{|F(M)|})$ be the order in which the faces of M are visited by Mesh2disk. Let D_j, for $j = 1$ to $|F(M)|$, be the simplicial complex after faces f_1 through f_i have been appended. We prove D_j is a disk by induction on j. For the base case, D_1 consists of a single face f_1 and hence is a disk. Now assume, D_{j-1} is a disk. When f_j is added to D_j by gluing edge-wise to f_i for some $i < j$, $|V(D_j)| = |V(D_{j-1})| + 1$, $|E(D_j)| = |E(D_{j-1})| + 2$ and $|F(D_j)| = |F(D_{j-1})| + 1$ (c.f. Figure 5.) Hence, $\chi(D_j) = \chi(D_{j-1})$. The number of boundary components remains the same. By the classification theorem of surfaces with boundary, D_j is a disk. □

Lemma 2. *For the input-output pair (M, D) of Mesh2disk, $|V(D)| = |F(M)| + 2$.*

Proof. Following the notation of Lemma 1, the claim follows by observing that D_j has one more vertex than D_{j-1}, for $j \geq 2$, and a straightforward induction on the number of faces of M. □

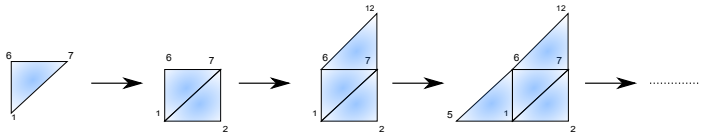

Fig. 5. Initial stages of Mesh2disk (Algorithm 1) constructing a polygonal schema of the torus mesh in Figure 4.

Lemma 3. *The algorithm Mesh2disk runs in $\mathcal{O}(|F(M)|)$ time.*

Proof. Mesh2disk is simply BFS as implied by the face-to-face adjacency list of M and thus takes $\mathcal{O}(|F(M)| + |E(M)|)$ time. By double counting of edges, $3|F(M)| = 2|E(M)|$. The claim follows. □

Each edge in the cut graph of a mesh determines exactly two boundary edges in the polygonal schema [5,9]. The boundary edges on D always come in pairs and if we glue all such pairs together we obtain the original mesh M. If e and e' are two boundary edges coming from the same edge in M, we say that e and e' are *partners*.

3.2 Removing Duplicate Edges on Disk Boundary

In principle, we can construct a Chinese postman tour on the polygonal schema D and map it back to an approximate Chinese postman tour on the input mesh M. However, an attempt to build a Chinese postman tour directly on D can repeat the cut graph edges on the mesh M three or four times since these edges appear twice as boundary edges of D. To guarantee that no edge of the mesh is traced more than twice, we instead find a Chinese postman tour on an embedded subgraph H of D with the following properties:

1. For any two partnered edges in D, exactly one of the two edges is in $E(H)$. Thus, the edge set of H has a one-to-one correspondence with the edge set of M.
2. H is a connected spanning subgraph of D, that is $V(H) = V(D)$.

After extracting such a spanning subgraph H, we can find a Chinese postman tour on H, which then maps to an approximate Chinese postman tour on M that visits the mesh edges at most twice. To extract H, we first identify two types of faces of D. A face in D is said to be a *peripheral face* of *type I* if it is bounded by only one boundary edge, and of *type II* if it is bounded by two boundary edges; see the boundary edges in the left-most image in Figure 6. Our algorithm, named Declone and presented as Algorithm 2, runs through the peripheral faces, identifying partnered edges and removing the clones along the way. For the polygonal schema of the torus in Figure 4, an intermediate output of Delcone is shown in the center in Figure 6 while the final output is shown on the right.

To prove Declone's output H satisfies the two properties listed in Section 3.2, we first define a new graph $X^\star$ which we refer to as the *cut-dual*.[5] The cut-dual is constructed by creating a vertex corresponding to each peripheral face in D, and adding an edge between two vertices u and v if and only if u's face contains an edge which has a partner

[5] $X^\star$ is the subgraph of the dual of M induced by the duals of the cut edges.

Algorithm 2: Declone: Remove clone edges on a polygonal schema's boundary.

Input: A polygonal schema D of a surface mesh M.
Output: An embedded subgraph H of D with the two properties listed in Section 3.2.

for *all faces* $f \in D$ **do**
  Mark f as not processed;
  if f *has a single boundary edge* **then** append f to F_I ;
  else if f *has two boundary edges* **then** append f to F_{II} ;
$H \leftarrow D$;
for *all faces* $f \in F_I$ **do**
  if f *is not processed* **then**
    $a \leftarrow$ a boundary edge of f;
    Remove a from H;
    Mark f as processed;
    $g \leftarrow$ the face which contains a's partner;
    while g *has two boundary edges* **do**
      $b \leftarrow$ the boundary edge in g different from a's partner;
      Remove b from H;
      Mark g as processed;
      $g \leftarrow$ the face which contains b's partner;
      $a \leftarrow b$'s partner;
    Mark g as processed;
for *all faces* $f \in F_{II}$ **do**
  if f *is not processed* **then**
    $g \leftarrow$ f;
    $a \leftarrow$ a boundary edge of g;
    while g *is not processed* **do**
      $b \leftarrow$ the other boundary edge of g;
      Remove b from H;
      Mark g as processed;
      $g \leftarrow$ the face which contains b's partner;
      $a \leftarrow b$'s partner;
return H

in v's face. For clarity, we refer to the vertices in $X^\star$ through their corresponding faces in D. Note that $X^\star$ has a maximum degree two since a peripheral face has at most two boundary edges. Hence, $X^\star$ is a disjoint union of connected components, each of which is either a path or a cycle. Each path starts and ends with a type I face but is otherwise composed of type II faces. Analogously, each cycle is completely composed of type II faces. With $X^\star$ in mind, we now prove the following lemmas about Declone.

Lemma 4. *Let e and e' be two partner boundary edges in a polygonal schema D of a surface mesh M, then exactly one of the two edges e or e' is removed in H. Hence, there is a one-to-one correspondence between $E(H)$ and $E(M)$.*

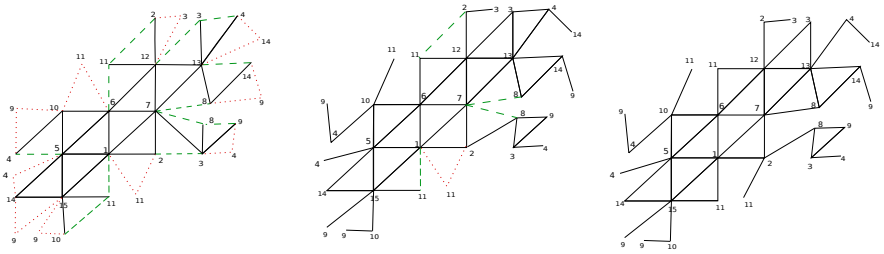

Fig. 6. Left, the 1-skeleton of the polygonal schema D of the torus mesh. Middle and right figures illustrate an intermediate output and the final output of Algorithm Declone, respectively. In the left and middle figures, the dashed edges highlight unprocessed type I peripheral faces while the dotted edges highlight unprocessed type II peripheral faces.

Proof. To show that exactly one of the edges e and e' is removed in H, consider the peripheral faces f and g that contain the edges e and e', respectively. By construction of the cut-dual $X^\star$, f and g appear in the same path or the same cycle component of $X^\star$.

Suppose f and g appear in a path P. Let s be the type I face of P appended first to F_I and let t be the other type I face in P. Suppose we orient P from s to t. Declone (Line 6 to 19) processes the faces in P from s to t . Suppose, w.l.o.g, f appears before g in the (oriented) P. If $f = s$, then e gets removed in Line 9, but e' is retained, whether g is a type II face (in the while loop), or it is a type I face (outside the while loop.) If $f \neq s$, then it gets processed in the while loop. Once again, e gets removed (Line 14) and e' gets retained whether we stay in the while loop or exit it in the next iteration. Analogously, e' gets removed and e gets retained when g appears before f in P.

Now suppose f and g appear in a cycle C of $X^\star$. Since C contains no type I face, none of the faces in C are processed before the third for loop (Line 19). Let s be the face of C which appears first in F_{II}. Orient C from s outward based on the selected edge a in Line 22. If f precedes g in the path along the oriented C starting from s, then f gets processed before g and e is removed while e' is retained. If g precedes f, e' is removed while e is retained. □

Lemma 5. *Declone computes a connected spanning subgraph H of the input D.*

Proof. Since no vertex is deleted in H, $V(H) = V(D)$. Hence, we only need to show that H is connected. Further noting that only a subset of the boundary edges of D get deleted, let us first analyze the state of the peripheral faces of D. In particular, we first show that for any peripheral face f in D, at most one boundary edge of f is deleted and the vertices of f remain connected in H.

Let u, v, w be the vertices of f and let $a = \{u, v\}, b = \{v, w\}, c = \{w, u\}$ be its edges. If f is a type I face, only one edge is a boundary edge and is potentially removed. If indeed the edge, w.l.o.g suppose a , is removed, its endpoints u and v remain connected through the path (u, c, w, b, v).

Now suppose f is a type II face, and w.l.o.g, assume that a and b are the boundary edges. Let g and g' be the two peripheral faces containing the partners of a and b, respectively. Reconsider the cut-dual $X^\star$ and the orientation of its paths and cycles (see

proof of Lemma 4.) In the oriented path or cycle, the order is either (g, f, g') or (g', f, g). In the first case, b is removed while a is retained while in the second case, the reverse holds. If a gets removed, then u and v remain connected through the path (u, c, w, b, v) while if b is removed, v and w are connected through (v, a, u, c, w).

We can now show that H is a connected graph. Since D is connected, there is a walk W between any vertices u, v. For every boundary edge of D that appears in W but is deleted in H, replace it with one of the paths described above. This results in a new walk between u and v in H, and thus H is connected. □

Lemma 6. *Declone runs in $\mathcal{O}(|F(M)|)$ time.*

Proof. Let p be the number of peripheral faces in D. Since the peripheral faces are a subset of the faces of D, $p \leq |F(D)| = |F(M)|$. The first for loop iterates $|F(D)| = |F(M)|$ times, each time consuming constant time. Note that checking face incidence takes constant time since each face is incident to three other faces. This also implies that the copy in Line 5 is linear in $|F(M)|$.

For the next two for loops, recall the proof in Lemma 4 and observe that each peripheral face gets processed once, either in a path or a cycle of $X^\star$. Hence, there are only p iterations of the total work done inside the while loops. The book-keeping, edge-removal and partner-checking can all be done in constant time, once again since each face is incident to three faces. □

3.3 Finding Non-crossing Chinese Postman Tours on the Polygonal Schema

By Lemma 5, Declone outputs a connected spanning subgraph H of the polygonal schema D. Since H is generally a non-Eulerian graph, cut-and-route proceeds by adding a minimal set of duplicate edges which converts H to an Eulerian multigraph H' using Edmonds' Blossom algorithm [7,19]. For the H graph shown on the right in Figure 6, the resulting Eulerian multigraph H' is shown on the left in Figure 7. To keep H' in the polygonal schema, the duplicate edges are added in the interior side of the polygonal schema. In principle, we can then compute an Eulerian tour on H' with Hierholzer's algorithm [17,19], but such a tour can generally have crossings, which complicates the analysis of the unknottedness of the tour.

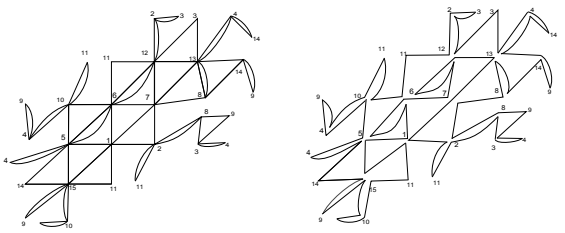

Fig. 7. Left, an Eulerian multigraph of H obtained by Edmonds algorithm. Right, a non-crossing Chinese postman tour on H, which then maps to an unknotted postman tour on the torus 1-skeleton.

To make the notion of crossing precise, note that a multigraph embedded on an orientable surface induces a local cyclic rotational order (fixed either clockwise or counterclockwise) of the incident edges of vertices. We say two pairs of consecutive edges in a closed-trail, all of which are incident to a common vertex, *cross* if the two pairs interleave with respect to the cyclic order of edges around the vertex. In other terms, a *crossing* is a quadruple of edges incident to a common vertex which can be grouped into two pairs according to their contiguity in the closed trail such that cyclically visiting edges around the vertex one alternates between the pairs. An Eulerian tour is said to be *non-crossing* if it does not contain any two crossing pairs of consecutive edges. Visually, a non-crossing Eulerian tour on a surface embedded multigraph can be drawn as a simple closed curve on the surface. In this sense non-crossing Eulerian tours detach to simple closed curves and are easier to analyze for unknottedness.

Abraham and Kotzig [1] as well as Grossman and Reingold [31] have shown that all Eulerian multigraphs embedded on a plane admit non-crossing Eulerian tours. More recently, Tsai and West [32] presented a technique to convert an arbitrary Eulerian tour on a multigraph embedded on a plane to a non-crossing one by a vertex-local re-splicing of the tour at crossing pairs.[6] Inspired by the proof of Grossman and Reingold [31], we introduce a linear-time algorithm in Lemma 7 to compute non-crossing Eulerian tours for multigraphs embedded on orientable surfaces. Briefly, the algorithm first computes an initial non-crossing closed-trail decomposition of the multigraph, and then iteratively resplices independent closed trails at each vertex, finally yielding a non-crossing Eulerian tour.

For a more precise description, we need the additional notion of a transition system of an Eulerian multigraph. A *transition system* [11] of an Eulerian multigraph is a set of partitions of edges, where each element of the set is a grouping (partitioning) of the incident edges of a vertex into (unordered) pairs. In a transition system, each edge is in two pairs corresponding to its' pairings at its' two end-points. There is a bijection between transition systems and closed-trail decomposition of Eulerian multigraphs [15]. In this setting, the pairs in the partitions of the transition system correspond to consecutive edges in the trails of the closed-trail decomposition. Given a transition system, we can compute the closed-trail decomposition in linear time by directionally following the pairings. The notion of non-crossing closed-trails extends naturally to closed-trail decompositions (and transition systems), in the sense that the closed trails are both self non-crossing and mutually non-crossing.

Lemma 7. *Let G be an Eulerian multigraph embedded on an orientable surface. There is an $\mathcal{O}(|V(G)| + |E(G)|)$ algorithm to compute a non-crossing Eulerian tour on G.*

Proof. As an input, assume that the embedded multigraph G is given as an adjacency list, describing for each vertex the list of the incident edges in the rotation order. The algorithm proceeds as follows. We first obtain an initial transition system by going through each vertex, and for each vertex, cycling around the incident edges in the rotation order and pairing the first with the second, the third with the fourth, etc. From

[6] Although stated only for Eulerian multigraphs embedded on a plane, Tsai and West's proof also holds for Eulerian multigraphs embedded on any surface since the resplicing occurs locally at vertices.

this initial transition system, we follow the pairings to compute an initial closed-trail decomposition. Next, for each vertex, we again cycle through the incident edges of the vertex and every instance where the current edge b is not in the same closed trail as the previous edge a, we pair a with b and a's old mate a_p with b's old mate b_p.

To prove the algorithm computes a non-crossing Eulerian tour, it suffices to prove that all edges are in the same non-crossing closed trail after all the vertices have been processed. We prove all edge are in the same closed trail by showing that after a vertex has been processed, we obtain a non-crossing closed-trail decomposition where all the edges incident to that vertex are in the same closed trail. After all vertices have been processed, this local criterion suffices to ensure distal edges are in the same closed trail because G is connected. For an alternative proof of the sufficiency of the local criterion, observe that since G is Eulerian, it has some Eulerian tour T'. Now suppose an edge e is in some closed trail T in the closed-trail decomposition. Starting from e, following the edges in T', we see that all the edges in T' are also in T since consecutive edges in T' are incident to a common vertex. Since T' is an Eulerian tour, all the edges in G are in T', and hence all edges in G are in T.

Hence, we just need to prove that after a vertex has been processed, we obtain a non-crossing closed-trail decomposition where all the local edges are in the same closed-trail. We prove this by induction on the incident edges of the vertex. In particular, we prove, for a local rotation index j (which runs from one up-to the degree of the vertex), the following two claims hold after the j-th edge has been processed: (1) The closed-trail decomposition is non-crossing, (2) The first j edges in the cyclic-order around the vertex are in the same closed-trail.

For the base case $j = 1$, (1) holds for the first processed vertex because in the initially computed transition system (and correspondingly closed-trail decomposition), the pairings are composed of neighboring edges and thus do not cross any other pairings. For the remaining vertices, (1) holds by induction on vertices. Claim (2) holds vacuously for $j = 1$.

Now suppose after the $(j-1)$-th edge has been processed we have a non-crossing closed-trail decomposition where all local edges are in the same closed-trail. If the $(j-1)$-th edge is in the same closed-trail as the j-th edge, nothing changes after the j-th edge is processed and both (1) and (2) still hold. Now suppose, j is not in the same closed-trail as $j-1$. Let b be the j-th edge and let a be the $(j-1)$-th edge, and let a_p and b_p be the mates of a and b before the re-pairing. To show that (1) holds, we only need to show that the new pairings $\{a, b\}$ and $\{a_p, b_p\}$ do not cross any of the unaltered pairings or each other. Indeed, the pairing $\{a, b\}$ is between neighboring edges and cannot cross any other pairing. To see that $\{a_p, b_p\}$ does not cross any other pairings, first note that b_p comes before a_p in the cyclic order around the vertex after a; that is, the cyclic order O is of the form $(\cdots, a, b, \cdots, b_p, \cdots, a_p, \cdots)$. This follows because $\{a, a_p\}$ does not cross $\{b, b_p\}$ by induction hypothesis. Now assume, for the sake of contradiction that $\{a_p, b_p\}$ crosses some other pairing $\{c, d\} \neq \{a, b\}$ at the vertex. Note that exactly one of c or d must be in between b_p and a_p in O since otherwise $\{c, d\}$ would not cross $\{a_p, b_p\}$. Without loss of generality, suppose c is the edge in between b_p and a_p in O. If d is between b and b_p, then $\{c, d\}$ crosses $\{b, b_p\}$ and if d is between a_p and a, then $\{c, d\}$ crosses $\{a, a_p\}$, in either case contradicting the induction hypothesis.

Hence, $\{c, d\}$ does not cross $\{a_p, b_p\}$ and claim (1) holds. For claim (2), since a and b are now paired, we only need to show that the re-pairing does not leave the edges from 1 to $j-1$ in different trails. This cannot be the case since breaking a closed trail at one pairing does not disconnect its' edges.

For the complexity, observe that, in both the computation of the initial decomposition and main processing, every vertex is processed once and every edge is checked at most twice. Hence, the $\mathcal{O}(|V(G)| + |E(G)|)$ time-complexity follows if we can show all the internal operations cost constant time. The initial pairing as well as re-pairing consumes constant time if we represent a transition system by maintaining, for each edge, the two mates of the edge at its two end-points. Checking whether two edges are in the same trail can be done in constant time if we associate with each edge, a pointer to their trail within the decomposition. That is, two pointers will point to the same trail if their corresponding edges are in the same trail. The pointers can be initialized in $\mathcal{O}(|E(G)|)$ time during the computation of the initial closed-trail decomposition. When re-pairing, updating the trail of a re-paired edge through its' pointer will simultaneously update the trails of all the edges in the same trail. □

A non-crossing Eulerian tour of H', and correspondingly a non-crossing Chinese postman tour of H, for our running example is shown on the right in Figure 7. Incorporating the algorithm presented in Lemma 7, cut-and-route outputs a non-crossing Chinese postman tour $\hat{T}$ on H which detaches to a simple closed curve on the polygonal schema D. Noting that, by Lemma 4, $\hat{T}$ is also a postman tour on the input mesh M, we now have the following theorems.

Theorem 1. *Cut-and-route outputs a postman tour which is detachable to the unknot on the input mesh M.*

Proof. Let C be the detached postman tour on M obtained by the cut-and-route algorithm. We prove that C is unknotted. By construction, C lies on the embedding of the polygonal schema D in $\mathbb{R}^3$. There is a homeomorphism h which maps the polygonal schema D, from its embedding in $\mathbb{R}^3$, to the standard disk on the plane. By homeomorphism restriction, h maps C to a simple closed curve C' on the plane. By the Jordan-Schönfflies' theorem [26], C' bounds a disk D'. The inverse of D' under h is a disk on the embedding of D whose boundary is C. A simple closed curve that bounds an embedded disk is an unknot. Hence, C is unknotted. □

Theorem 2. *Cut-and-route outputs an unknotted postman tour that visits any edge of the input mesh M at most twice. Moreover, it is a $\frac{5}{3}-$approximation algorithm for **UCPT**.*

Proof. Let $\hat{T}$ be the outputted non-crossing Chinese postman tour on H. The first claim follows because a Chinese postman tour visits every edge at most twice [7], and by Lemma 4, the edges of H are in one-to-one correspondence with the edges of M.

For the second claim, observe that the set of edges added to H to construct the Eulerian multigraph H' form a forest on a subset of $V(H)$. Indeed, if such a graph has a cycle, the edges in the cycle can be removed from H' while keeping it Eulerian. With the cycle removed, an Eulerian tour on H' yields a postman tour with less length than $\hat{T}$, thus contradicting the minimality of $\hat{T}$. By the relationship between the number

of vertices and number of edges of a forest, there are at most $|V(H)| - 1$ extra edges in H'. By Lemmas 2 and 5, $|V(H)| = |V(D)| = |F(M)| + 2$. By double counting of edges, $|F(M)| = \frac{2}{3}|E(M)|$. Thus, at most $\frac{2}{3}|E(M)| + 1$ edges are repeated. Since any Chinese postman tour visits every edge, $|E(M)|$ is a lower-bound on the optimal unknotted Chinese postman tour. The approximation factor thus follows. □

Theorem 3. *For an input mesh M, cut-and-route runs in $\mathcal{O}(|F(M)|^3)$ time.*

Proof. For the analysis, note the relations $|V(H)| = |V(D)| = |F(M)| + 2$, $|E(H)| = |E(M)|$ implied by Lemmas 2, 5 and 4. Also note that $|V(H'| = |V(H)|$ and $|E(H')| \leq 2 * |E(H)|$ from Edmonds' algorithm [7,19] and the fact that $2|E(M)| = 3|F(M)|$ by double counting. Mesh2disk and Declone were shown to run in $\mathcal{O}(|F(M)|)$ time in Lemmas 3 and 6. Edmonds' algorithm to convert H to the Eulerian counterpart H' runs in $\mathcal{O}(|V(H)|^3) = \mathcal{O}(|F(M)|^3)$ time [7,19]. It is easy to check (c.f. Supplementary methods in [2]) that we can compute, in no more than cubic time, the local rotation of edges at vertices for H (and in turn of H') from the polygonal schema description and the deleted edges of H. By Lemma 7, computing a non-crossing Eulerian tour on H', or equivalently finding a non-crossing Chinese postman tour on H, takes $\mathcal{O}(|V(H')| + |E(H')|) = \mathcal{O}(F(M))$ time. Hence, all the modules of cut-and-route run in no more than cubic-time with respect to the number of faces of the input mesh M. □

4 Conclusions and Future Work

Eulerian tours have previously featured in experimental and theoretical considerations of DNA [3,8,18,25,35] and protein self-assembly [14,21]. Similarly, topological constraints have been implicitly considered in previous works [3,6]. Here, we formally investigated an unknottedness constraint of a circular strand's routing on triangulated higher-genus surfaces, mostly within the design-framework of Benson et al. [3]. We presented a cubic-time algorithm to compute unknotted approximate Chinese postman tours on such surfaces.

There are numerous theoretical questions available within the prescribed theory of unknotted Chinese postman tours. In the specified setting, the complexity of finding unknotted Chinese postman tours (**UCPT** defined in Section 2.3) on surface meshes remains. Simultaneously, further improvements to the approximation with respect to the approximation factor and run-time can also be pursued. More generally, **UCPT** on straight-line graph embeddings in $\mathbb{R}^3$ can be studied with an aim to design arbitrary non-manifold wireframe structures.

On the experimental side, it remains to be seen whether the current approach is viable, especially with the implied generality. First, unknottedness, while necessary, is likely insufficient for knotted surface embeddings such as the one depicted in Figure 3. As evident in such instances, an unknotted routing can still have a self-threading of the strand through loops. Although it has been previously shown [22] that such self-threadings are attainable through careful design of the folding pathway, such designs may be unlikely to fold purely from thermodynamic optimization.

References

1. Abrham, J., Kotzig, A.: Construction of planar Eulerian multigraphs. In: Proc. Tenth Southeastern Conf. Comb., Graph Theory, and Computing. pp. 123–130 (1979)
2. Benson, E., Mohammed, A., Bosco, A., Teixeira, A.I., Orponen, P., Högberg, B.: Computer-aided production of scaffolded DNA nanostructures from flat sheet meshes. Angewandte Chemie International Edition 55(31), 8869–8872 (2016)
3. 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)
4. Chen, J., Seeman, N.C.: Synthesis from DNA of a molecule with the connectivity of a cube. Nature 350(6319), 631 (1991)
5. Dey, T.K., Schipper, H.: A new technique to compute polygonal schema for 2-manifolds with application to null-homotopy detection. Discrete & Computational Geometry 14(1), 93–110 (1995)
6. 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)
7. Edmonds, J., Johnson, E.L.: Matching, Euler tours and the Chinese postman. Mathematical Programming 5(1), 88–124 (1973)
8. Ellis-Monaghan, J.A., Pangborn, G., Seeman, N.C., Blakeley, S., Disher, C., Falcigno, M., Healy, B., Morse, A., Singh, B., Westland, M.: Design tools for reporter strands and DNA origami scaffold strands. Theoretical Computer Science (2016)
9. Erickson, J., Har-Peled, S.: Optimally cutting a surface into a disk. Discrete & Computational Geometry 31(1), 37–59 (2004)
10. Euler, L.: Solutio problematis ad geometriam situs pertinentis. Commentarii Academiae Scientiarum Petropolitanae 8, 128–140 (1741)
11. Fleischner, H.: Eulerian graphs and related topics, vol. 1. Elsevier (1990)
12. Floater, M.S.: Parametrization and smooth approximation of surface triangulations. Computer Aided Geometric Design 14(3), 231–250 (1997)
13. Goodman, R.P., Schaap, I.A., Tardin, C.F., Erben, C.M., Berry, R.M., Schmidt, C.F., Turberfield, A.J.: Rapid chiral assembly of rigid DNA building blocks for molecular nanofabrication. Science 310(5754), 1661–1665 (2005)
14. Gradišar, H., Božič, S., Doles, T., Vengust, D., Hafner-Bratkovič, I., Mertelj, A., Webb, B., Šali, A., Klavžar, S., Jerala, R.: Design of a single-chain polypeptide tetrahedron assembled from coiled-coil segments. Nature Chemical Biology 9(6), 362–366 (2013)
15. Gross, J., Yellen, J.: Handbook of Graph Theory. Discrete Mathematics and Its Applications, CRC Press (2004)
16. He, Y., Ye, T., Su, M., Zhang, C., Ribbe, A.E., Jiang, W., Mao, C.: Hierarchical self-assembly of DNA into symmetric supramolecular polyhedra. Nature 452(7184), 198–201 (2008)
17. Hierholzer, C., Wiener, C.: Über die Möglichkeit, einen Linienzug ohne Wiederholung und ohne Unterbrechung zu umfahren. Mathematische Annalen 6(1), 30–32 (1873)
18. Jonoska, N., Seeman, N.C., Wu, G.: On existence of reporter strands in DNA-based graph structures. Theoretical Computer Science 410(15), 1448–1460 (2009)
19. Jungnickel, D., Schade, T.: Graphs, networks and algorithms. Springer (2008)
20. Ke, Y., Ong, L.L., Shih, W.M., Yin, P.: Three-dimensional structures self-assembled from DNA bricks. Science 338(6111), 1177–1183 (2012)
21. Klavzar, S., Rus, J.: Stable traces as a model for self-assembly of polypeptide nanoscale polyhedrons. MATCH Commun. Math. Comput. Chem 70, 317–330 (2013)
22. Kočar, V., Schreck, J.S., Čeru, S., Gradišar, H., Bašić, N., Pisanski, T., Doye, J.P., Jerala, R.: Design principles for rapid folding of knotted DNA nanostructures. Nature Communications 7 (2016)

23. Lee, J.: Introduction to topological manifolds, vol. 940. Springer Science & Business Media (2010)
24. Lickorish, W.R.: An introduction to knot theory, vol. 175. Springer Science & Business Media (2012)
25. Morse, A., Adkisson, W., Greene, J., Perry, D., Smith, B., Ellis-Monaghan, J., Pangborn, G.: DNA origami and unknotted A-trails in torus graphs. arXiv preprint arXiv:1703.03799 (2017)
26. Rolfsen, D.: Knots and links, vol. 346. American Mathematical Soc. (1976)
27. Rothemund, P.W.: Folding DNA to create nanoscale shapes and patterns. Nature 440(7082), 297–302 (2006)
28. Rothemund, P.W., Papadakis, N., Winfree, E.: Algorithmic self-assembly of DNA Sierpinski triangles. PLoS Biol 2(12), e424 (2004)
29. Seeman, N.C.: Nucleic acid junctions and lattices. Journal of Theoretical Biology 99(2), 237–247 (1982)
30. Shih, W.M., Quispe, J.D., Joyce, G.F.: A 1.7-kilobase single-stranded DNA that folds into a nanoscale octahedron. Nature 427(6975), 618–621 (2004)
31. Singmaster, D., Grossman, J.W.: E2897. The American Mathematical Monthly 90(4), 287–288 (1983)
32. Tsai, M.T., West, D.B.: A new proof of 3-colorability of Eulerian triangulations. Ars Mathematica Contemporanea 4(1), 73–77 (2011)
33. Veneziano, R., Ratanalert, S., Zhang, K., Zhang, F., Yan, H., Chiu, W., Bathe, M.: Designer nanoscale DNA assemblies programmed from the top down. Science 352(6293), 1534–1534 (2016)
34. Wei, B., Dai, M., Yin, P.: Complex shapes self-assembled from single-stranded DNA tiles. Nature 485(7400), 623–626 (2012)
35. Wu, G., Jonoska, N., Seeman, N.C.: Construction of a DNA nano-object directly demonstrates computation. Biosystems 98(2), 80–84 (2009)
36. Zheng, J., Birktoft, J.J., Chen, Y., Wang, T., Sha, R., Constantinou, P.E., Ginell, S.L., Mao, C., Seeman, N.C.: From molecular to macroscopic via the rational design of a self-assembled 3D DNA crystal. Nature 461(7260), 74–77 (2009)