Algorithmic Design of Cotranscriptionally Folding 2D RNA Origami Structures*

Abdulmelik Mohammed[1], Pekka Orponen[1], and Sachith Pai[1]

Aalto University, Department of Computer Science
FI-00076 Aalto, Finland
abdulmelik.mohammed@aalto.fi,
pekka.orponen@aalto.fi,
sachith.pai@aalto.fi

Abstract. We address a biochemical folding obstacle of "polymerase trapping" that arises in the remarkable RNA origami tile design framework of Geary, Rothemund and Andersen (Science 2014). We present a combinatorial formulation of this obstacle, together with an optimisation procedure that yields designs minimising the risk of encountering the corresponding topological trap in the tile folding phase. The procedure has been embedded in an automated software pipeline, and we provide examples of designs produced by the software, including an optimised version of the RNA smiley-face tile proposed by Geary and Andersen (DNA 2014).

Keywords: RNA origami, RNA tiles, RNA nanotechnology, Rational design, Cotranscriptional folding, Grid graphs, Spanning trees

1 Introduction

Following the introduction of Paul Rothemund's DNA origami technique in 2006 [14], the research area of DNA nanotechnology [15] has made rapid progress in the rational design of highly complex 2D and 3D DNA nanostructures and their applications [10, 12, 13, 16, 18]. In the past few years, there has also been increasing interest in using RNA, rather than DNA, as the fundamental construction material for similar purposes [5, 7–9]. One great appeal of this alternative is that while the production of designed DNA nanostructures typically proceeds by a multi-stage laboratory protocol that involves synthesising the requisite nucleic acid strands and hybridising them together in a thermally controlled process, RNA nanostructures can in principle be produced in quantity by the natural process of polymerase transcription from a representative DNA template, isothermally at room temperature, *in vitro* and eventually *in vivo*.

The challenge in this approach, however, is that in contrast to DNA, RNA characteristically exists in single-stranded form, and the varied 3D conformations

* Research supported by Academy of Finland grant 311639, "Algorithmic Designs for Biomolecular Nanostructures (ALBION)".

of RNA molecules are the result of a given strand folding upon itself in tertiary structures whose formation is quite difficult to predict and control algorithmically. Nevertheless, in the emerging field of RNA nanotechnology, there have also been several approaches to the rational design of RNA nanostructures [7, 9]. For instance, in "RNA tectonics" [1, 3, 17], well-characterised elementary structural modules are linked together by connector motifs to form intricate 2D and 3D complexes.

On the other hand, the approach of *de novo* algorithmic structure design, which has been so successful in the case of DNA origami, has been less explored in the context of RNA, most likely because of the higher complexity of RNA's single-stranded folding kinetics. One notable exception has been the work of Geary, Rothemund and Andersen [5], which presents an approach to designing 2D "RNA origami tiles" by a systematic scheme of intra-structure couplings of collinear helical stem segments by crossover and kissing-loop motifs.

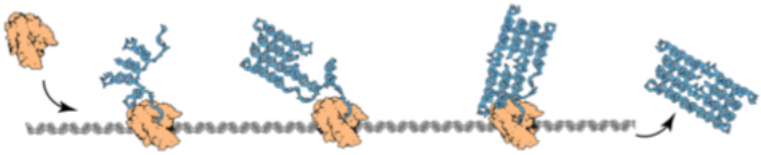


Fig. 1: Cotranscriptional folding of a 2D RNA origami tile from a DNA template, mediated by an RNA polymerase enzyme. Reprinted with permission from [4].

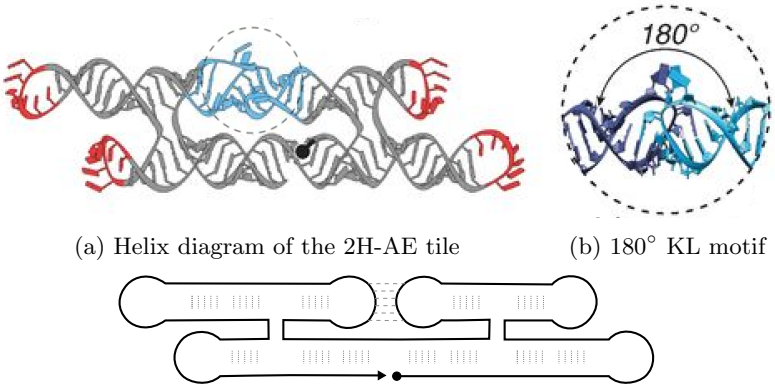
Geary et al. [5] also demonstrate experimentally that the designed tiles can be folded in the laboratory both by a heat-annealing protocol from prefabricated RNA strands, and by a cotranscriptional protocol, whereby the RNA strand folds into its 2D conformation concurrently to being transcribed from its DNA template by an RNA polymerase enzyme (Figure 1).

While such cotranscriptional folding of *de novo* designed RNA nanostructures is a remarkable achievement, there appear to be some challenges in extending the methodology of [5] to bigger and more complex structures, related in particular to risks of kinetic and topological traps in the folding process. We shall discuss in this paper a systematic design approach that addresses one potentially significant topological obstacle which we call *polymerase trapping*. This involves the cotranscriptional folding of the RNA strand proceeding in such a way that the design's intra-structure kissing-loop interactions block its downstream helices from forming while the structure is still coupled to its large polymerase-DNA template complex.

In the following, Section 2 discusses the basic structure of the RNA origami tiles from [5] and introduces the polymerase trapping problem. Section 3 then presents a more abstract view of origami tiles as renderings of 2D grid graphs, and how minimising the risk of polymerase trapping can be formulated as a combinatorial optimisation task in this general framework. Section 4 discusses

a branch-and-bound solution method for this task, which requires a somewhat nontrivial search in the very large space of spanning trees of a given grid graph. Section 5 outlines our software pipeline that leads from a bitmap design of a targeted 2D pattern to a secondary-structure description of an RNA strand that would fold to render that pattern as a generalised tile, with minimal risk of polymerase trapping.[1] This Section also contains some examples of (almost) completely trap-free designs, including a smiley-face design embedded on 14×6 -grid, similar to the one presented in article [6]. Section 6 concludes with some general observations and further challenges.

2 RNA Tiles, Cotranscriptional Folding and Polymerase Trapping



(a) Helix diagram of the 2H-AE tile (b) 180° KL motif

(c) Strand-path diagram of the 2H-AE tile

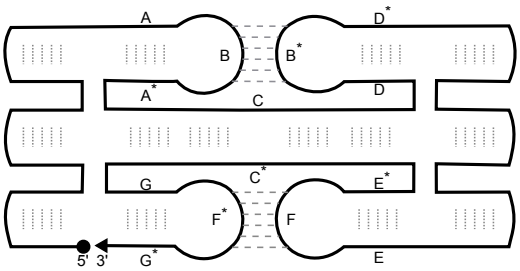
Fig. 2: (a) A helix-level diagram of the 2H-AE RNA origami tile from [5]. (b) The 180° kissing loop motif used as intra-structure connector. (c) A strand-path schematic of the tile. Figures (a) and (b) adapted with permission from [5].

We start our discussion by considering the structure of the 2H-AE tile, the simplest design from article [5]. Figure 2a presents a helix-level diagram of the 3D structure of this molecule. The 5' end of the RNA strand, marked here with a black dot, is located in the middle of the lower helix. From there the strand winds towards the right end of the diagram, creates a hairpin loop (marked in red), crosses to the upper helix, creates another hairpin loop etc. The most interesting

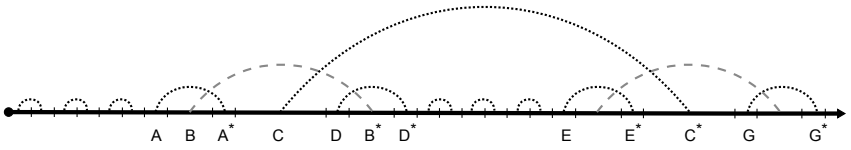
[1] We are currently working on the challenge of transforming the secondary-structure descriptions to actual RNA sequences, but lab-proof sequence design is a nontrivial task, and validating that the generated sequences really fold as intended requires experimental work.

part of the design is the kissing-loop motif (marked in blue) in the middle of the upper helix. This is a naturally occurring (dimerization initiation site of HIV-1 RNA) arrangement of two antiparallel RNA hairpin loops that hybridise together trans-helically to form a very precise 180° coupling between their respective hairpins: geometrically this is almost as if the helix constituting the stem of one hairpin continued into the other, even though there is no continuity in the strand. Figure 2b displays a slightly expanded view of this motif.

Figure 2c exhibits a more abstract strand-path diagram of the structure. Here the vertical dotted lines indicate the intra-helical stem pairings, and the dashed horizontal lines the trans-helical kissing-loop interactions. The 5' end of the strand is marked with a black dot and the 3' end with an arrowhead.



(a) Strand-path diagram of a 3H-AE tile



(b) Arc diagram of the 3H-AE tile

Fig. 3: (a) Strand path diagram of a 3H-AE tile. (b) A domain-level arc diagram of the 3H-AE tile.

Let us then consider the design of a 3H-AE tile, an extension of the 2H-AE tile with a third helical layer, and using the specific strand-path routing outlined in Figure 3a. One could also route the strand and arrange the kissing-loop connections differently for the same high-level 3×2 tile scheme (3 horizontal helices, 2 vertical cross-over seams), and we will return to this issue in Section 3. But for now let us focus on the specific H-like design shown in Figure 3a.

In Figure 3a, each main domain of the strand constituting the tile is labelled with a capital letter, and its complementary domain with the same letter followed by an asterisk. Figure 3b presents an arc diagram that outlines the pairings between these domains: the intervals between tick marks correspond to the respective strand segments, stem domain pairings are indicated with dotted

arcs and kissing loop interactions with dashed arcs. Note that compared to the 2H-AE tile from [5], our 3H-AE design has been simplified so that the perimeter kissing loops (denoted by red in Figure 2a), which are used to connect tiles to each other in [5], have been replaced by simple nonpairing tetraloop "caps".

Let us then consider how a cotranscriptional folding process for the 3H-AE tile structure presented in Figure 3 might proceed. Instead of thinking of the RNA strand being spooled out of the polymerase starting at the 5' end and folding as the appropriate base pairings become available, it may be easier to visualise the large polymerase-DNA template complex as traversing the 5'-3' strand route outlined in Figure 3a and generating the bases as it goes. Generating the A and B strand segments is uneventful, and the RNA strand stays linear until sometime after the A^* segment has been generated. (In reality of course several transient nonspecific pairings will arise during the folding process, but we are ignoring these in this simplified discussion.) Then segment A gets paired to segment A^*, D to D^*, the kissing loop B–B^* closes etc.

Consider now what happens when the polymerase reaches domain C^* which should constitute a double-strand helix with domain C by winding strand segment C^* around C. If kissing loop B–B^* has already closed, the strand with the big polymerase-DNA complex coupled to it cannot achieve this, since the kissing-loop pairing is blocking the pathway.

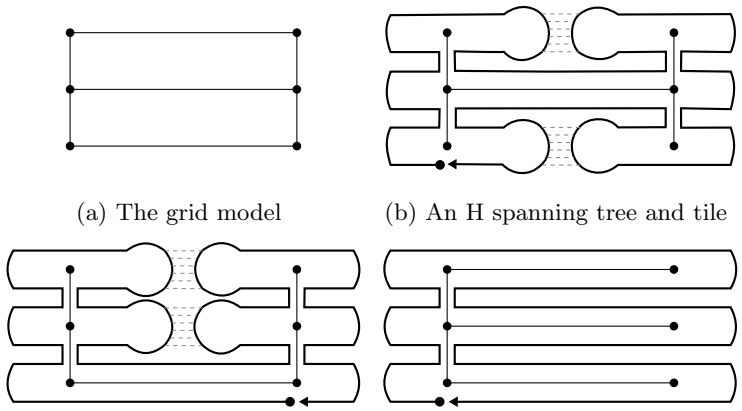
This topological folding obstacle of "polymerase trapping" is briefly addressed by Geary and Andersen in article [6] (Section 4.4), which discusses the technical design principles of RNA origami tiles. However, this article does not explain the background of this design constraint in any detail or formulate it in a general way. (The authors kindly explained these issues in a personal discussion.)

Viewed more closely, the significance of the polymerase trapping obstacle depends on the relative timescales of kissing loop formation and the speed of polymerase transcription. (In a purely combinatorial sense, the problem arises already in the 2H-AE tile design of Figure 2a, but there the time from kissing loop formation to the completion of the transcription is apparently so short that the issue does not significantly affect the experimental results.) This can be understood more clearly by considering the situation in the representative arc diagram: in the case of the 3H-AE tile, the problem is created by the long forward stem pairing C–C^* that emerges from inside the kissing loop pairing B–B^*. The longer the arc, the more time the enclosing kissing loop has to close, and the higher the likelihood that the folding process gets trapped by this obstacle.

In Section 3, we formulate the goal of minimising the risk of polymerase trapping as a design objective for tile design, and in Section 4 we discuss a computational approach to optimising this objective.

3 Tiles, Grids and Spanning Trees

In this Section, we introduce a combinatorial model for designing 2D RNA shapes, presented here for rectangular shapes and discussed in a more general framework in Section 4.



(a) The grid model (b) An H spanning tree and tile

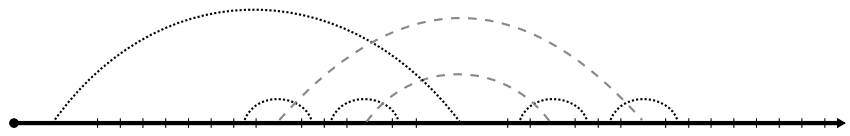
(c) A U spanning tree and tile (d) An E spanning tree and tile

Fig. 4: (a) 3 x 2 grid model for a 3-helix tall, ∼2-turn wide RNA rectangular shape, and (b)–(d) three tiles derived from three different spanning trees of the grid. The tiles are formed by routing the RNA strands around the spanning tree and bulging out kissing hairpin loops in towards non-spanning tree edges. In (b)–(d), the thick outer paths indicate the tiles' strand routings, the thin internal schematics outline the spanning trees of the grid, and the dashed horizontal lines in between the loops indicate kissing loop interactions.

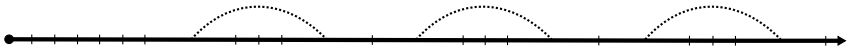
In our combinatorial model, we represent an M-helix tall, (N x u)-turn wide rectangular shape (tile) by an M x N grid. We assume the vertical dimension of the target shape to be a multiple of the diameter of an RNA A-helix (∼2.3 nm) and the horizontal dimension to be approximately a multiple u of A-helical turns (∼3.2 nm), where $u \geq 1$ is the minimum number of full-turns needed to implement an HIV-1 DIS type 180° kissing loop complex. For instance, the 3H-AE tile sketched in Figure 3 implements a rectangular shape derived from the 3 x 2 grid illustrated in Figure 4a. Correspondingly, the 2H-AE tile design by Geary et al. [5] in Figure 2a could be rendered from a 2 x 2 grid model; the four perimeter hairpin domains flanking the crossovers would then constitute an approximation error in the horizontal dimension.

Having employed a grid to model a rectangular shape, we aim to render the horizontal edges of the grid as either continuous A-helical stem domains, or as kissing-loop complexes, and a selected set of vertical edges as crossover locations. Note that since the vertical edges correspond to potential crossover locations, they essentially have zero length, even though they are presented, for the sake of clarity, with non-zero length in the schematics. The set of edges corresponding to the helical domains and the crossover locations are selected based on a spanning tree of the grid graph.[2]

[2] A *spanning tree* of a graph is a cycle-free subset of the graph that includes all the vertices of the graph [2, Chapter 23].



(a) Arc diagram of the U tile



(b) Arc diagram of the E tile

Fig. 5: Arc diagrams of the U and E tiles from Figure 4. Dotted arcs indicate stem pairings while dashed arcs show kissing loop pairings. The arc diagram in (a) reveals that the horizontal spanning tree edge of the U tile has no cost since the corresponding long stem pairing only crosses the kissing loops in the backward direction. Hence, the U tile only has a trivial cost due to the second hairpins of the kissing loops. The arc diagram in (b) shows that the E tile has a zero cost since it has no kissing loops. For clarity, the stem pairing arcs of the perimeter stem loops have been left out in both (a) and (b).

Accordingly, in order to design an RNA tile corresponding to the input shape, a spanning tree of the grid model is first computed and the single stranded RNA strand is routed twice around this tree.[3] Such a routing pairs distal segments of the RNA strand in an antiparallel fashion on the spanning tree edges, thus making it suitable for rendering horizontal spanning tree edges as A-helical domains and the vertical spanning tree edges as crossovers. Next, at every non-spanning tree horizontal edge, two hairpins are spliced into the strand routing at the edge's endpoints such that the hairpin loops kiss at the centre of the edge. To ensure every crossover is flanked by helical arms, short stems capped with inactive tetraloops are finally spliced to the routing at perimeter vertices, with the tetraloops facing horizontally outward. Three different tiles derived in such a manner from three different spanning trees of the 3 x 2 grid in Figure 4a are shown in Figures 4b, 4c and 4d. We refer to these three tiles as the "H", "U" and "E" tiles based on the resemblance of their associated spanning trees to the respective Latin letters. (The H tile is the 3H-AE example from Section 2.)

After generating a tile from a spanning tree, we can linearise its strand routing to an arc diagram and investigate it for cotranscriptional polymerase trapping. Note that the pairings in the tile, and correspondingly the arcs in the arc diagram, are determined by the strand routing; in particular, we place short arcs corresponding to the stems of the tetraloop capped perimeter hairpins, long-range stem arcs corresponding to the long-range A-helix stem pairings on the

[3] This standard graph algorithm technique is discussed e.g. in [2, Section 35.2].

spanning tree edges, and long-range kissing loop arcs corresponding to the kissing loop complexes on the non-spanning tree edges. The arc diagram of the H tile is shown in Figure 3b, while those of the U and E tiles are shown in Figure 5.

Recall that cotranscriptional polymerase trapping is a risk if there is a stem-pairing arc crossing a kissing-loop arc in the forward direction. Moreover, the trapping is more likely if the later segment of the stem pairing (e.g. segment C^* in Figure 3b) is transcribed much later than the second hairpin loop of the kissing loop (e.g. segment B^* in Figure 3b.) Hence, in case a stem pairing crosses a kissing loop in the forward direction, we associate a cost to the stem pairing proportional to the strand-distance between the second hairpin loop of the kissing loop and the stem pairing's second segment. If a stem pairing crosses multiple kissing loops, we associate with it the maximum cost over all the kissing loops it crosses. In this formulation, a stem pairing which does not cross any kissing-loop arc in the forward direction will have zero cost. For instance, the central stem pairing of the H tile (Figure 4b) has non-zero cost because it crosses the top kissing loop in the forward direction (cf. Figure 3b), but the bottom stem pairing of the U tile (Figure 4c) has zero cost since it crosses neither kissing loop in the forward direction (cf. Figure 5a). Also note that the stems of tetraloop capped perimeter stem loops have zero cost since they cross no kissing loops.

We set the cost of a tile to be the maximum over all costs of its stem pairings. Note that since the stem of the second hairpin of every kissing loop complex (e.g. stem D–D^* in Figure 3b) crosses the kissing loop (e.g. KL B–B^* in Figure 3b), every tile with a kissing loop has this trivial non-zero cost. In this regard, only the E tile (Figure 4d) has zero polymerase trapping cost (compare its arc diagram in Figure 5b with the other tiles' arc diagrams). Nevertheless, the U tile has a cost no more than the trivial hairpin stem cost since the only long range stem pairing, which corresponds to the horizontal spanning tree edge, has zero cost (cf. Figure 5a). In contrast, the H tile has non-trivial cost because the stem pairing on the spanning tree edge crosses the upper kissing loop (cf. Figure 3b). Even though the U tile thus technically has slightly larger cost than the E tile, it is more likely to stay well-formed than the E tile, due to its two-crossovers-per-row design that limits rotational flexibility compared to the single crossovers of the E tile. Hence our tile design scheme always imposes this constraint.

Note that the cost of a stem pairing depends on the 5' to 3' routing direction since the cost definition involves the crossing of a kissing-loop arc in the forward direction. In this regard, the main stem pairing of the U tile would have had a non-zero cost if the routing direction was reversed. Indeed, if the transcription direction was reversed in the arc diagram of Figure 5a, the stem pairing would have crossed both kissing-loop arcs in the forward direction. Furthermore note that, given a fixed spanning tree, the cost of a routing depends also on the starting point of the routing. For instance, starting a clockwise routing at the lowest left vertex of the U spanning tree (Figure 4c) would have yielded a non-trivial cost in the resulting tile because the spanning-tree-edge stem pairing would then have had a non-trivial cost. In particular, since the upper segment of the stem pairing would have preceded the complementary lower segment in the

linearisation to an arc diagram, the stem-pairing arc would have crossed both kissing-loop arcs in the forward direction. Since there are an infinite number of possible starting points, we limit routings to only start at vertices. Given the above two considerations, we associate with a spanning tree the minimum cost among all the possible combinations of starting points and directions (clockwise or counterclockwise).

4 Search Algorithm

In principle, we can develop the search for good spanning trees on arbitrary finite connected subgraphs of the infinite rectangular grid. To model reasonable 2D RNA shapes, we however limit our attention to subgraphs corresponding to bitmap shapes carved from the infinite grid (see Figure 6c). In particular, we shall consider the input to our algorithm to be a finite subgraph derived from a finite set of connected pixels (faces) of the infinite grid; we consider two pixels to be connected if there is a common vertex bounding both pixels. The input is then the set of vertices and edges bounding the selected pixels. To build RNA tiles out of such partial grids, we follow the same procedure as in the case of rectangular shapes (cf. Section 3), except that in this case, every vertex in the partial grid which only has one horizontal edge incident to it will be considered a perimeter/boundary vertex and will be flanked with a tetraloop capped stem loop in the missing horizontal edge (see e.g. the vertices bounding the eyes of the smiley-face in Figure 7).

Finding a good strand routing, i.e. one that is least likely to cause cotranscriptional polymerase trapping, entails searching through a large number of possible spanning trees of the input grid. For instance, even in the relatively small 6 x 6 complete grid, the number of spanning trees is approximately $3.2 * 10^{15}$ [11]. To effectively manage such a large search space, we developed a search procedure (Algorithm 1) that applies a branch-and-bound search on the spanning tree space of the underlying grid graph of the given shape. The branch-and-bound process conceptually performs an exhaustive search of all spanning trees, but prunes the search paths based on lower bounds evaluated from partial solutions, which in this case, correspond to trees spanning an incomplete set of vertices.

The algorithm's branch-and-bound search tree is based on binary choices for edges. At each step, the algorithm selects an edge and decides whether to include or exclude this edge (Lines 15 and 18); two branches corresponding to this decision are generated in the search tree. The choice edge is selected at random from the list of edges adjacent to the current spanning tree, i.e. to the current partial solution (Line 14 of Algorithm 1). To bound the search tree effectively, we use the cost of this spanning tree as the lower bound for all spanning trees which can be extended from it in the current search path. Note that the search process here evolves a single partial spanning tree to eventually find a tree that spans the complete target shape. This structuring of the search tree, combined with the bounding mechanism, makes it possible to find minimum-cost spanning trees for large designs such as the smiley-face in Figure 7. Alternatively, one could

Algorithm 1 Find a spanning tree that minimises risk of polymerase trapping

Input: A grid graph G modelling a 2D shape
Output: A minimum cost spanning tree

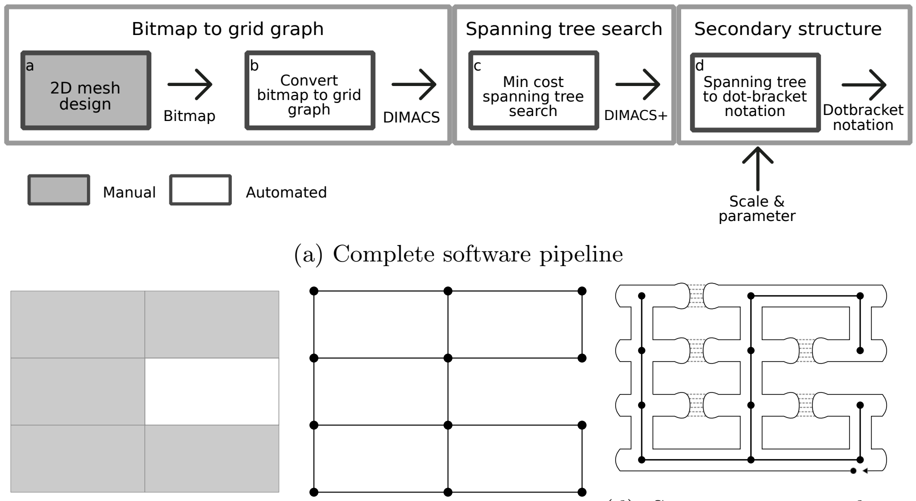
```
 1: best_tree ← Randomly generated spanning tree of G
 2: min_cost ← Cost(best_tree)
 3: RECURSIVE SEARCH(G, G, some vertex v of G)
 4: return best_tree
 5:
 6: procedure RECURSIVE SEARCH(G, residual, tree)
 7:     if tree is a valid spanning tree of G then
 8:         if Cost(tree) < best_cost then
 9:             min_cost ← Cost(tree)
10:             best_tree ← tree
11:             return
12:         end if
13:     end if
14:     new_edge ← Select from residual a random edge which is adjacent to tree but
    does not create a cycle when added to tree.
15:     if Cost(tree ∪ new_edge) < min_cost then
16:         Recursive search(G, residual − new_edge, tree ∪ new_edge)
17:     end if
18:     if new_edge is not a cut edge in residual then
19:         Recursive search(G, residual − new_edge, tree)
20:     end if
21: end procedure
```

have decided on arbitrary edges instead of edges adjacent to the current tree. However, this entails growing forests of trees as partial solutions, and leads to several difficulties in obtaining a lower-bounding function for the search process.

Efficient search through branch-and-bound search is possible because of the monotonically increasing cost function. Recall that the cost is incurred by stem-pairing arcs crossing kissing-loop arcs in the forward direction in the arc diagram representing the routing. When an edge is selected to extend the tree, it only adds a small segment to be spliced into the arc diagram of a routing around the tree. Clearly, this can never decrease the strand-distance of the forward crossing arcs. Therefore, adding edges to any tree can only increase the cost of the tree. The algorithm also prevents the possibility of a vertex from not being spanned through a connectivity check before the exclusion of an edge (lines 18-20). This connectivity check ensures the graph does not become disconnected as the result of an edge exclusion.

5 Design Pipeline and Examples

We have integrated our spanning tree search algorithm into a software design pipeline for generating RNA tiles from 2D meshes. The pipeline, along with a representative example, is presented in Figure 6. The design process starts in



(a) Complete software pipeline

(b) Bitmap design

(c) Grid graph

(d) Spanning tree with a min cost strand-routing

```
(((((((((((((((((((((((((((((((((((((((((((((((((((....)))))(((((....)))))(((((....)))))(((((....)))))((((((..
[[[[[[..))))))((((((..[[[[[[..))))))((((((..[[[[[[..))))))))))))))))))))))))))))))(((((((..]]]]]]..)))))))
(((((((..]]]]]]..)))))))(((((((..]]]]]]..)))))))((((((((((((((((((((((((((((....)))))(((((....)))))((((((..
[[[[[[..)))))))))))))))))))))))))))))))))))(((((((..]]]]]]..)))))))((((((..[[[[[[..))))))))))))))))))))))))))))))
(((((((..]]]]]]..)))))))(((((....)))))(((((....)))))
```

(e) The final secondary structure including kissing loops

Fig. 6: Our software pipeline for designing 2D RNA shapes.

a custom 2D mesh design tool we developed, whereby the user first sets the dimensions of the grid and henceforth selects a set of pixels constituting the target shape. After exporting the mesh as a bitmap, the user can run a script to generate an RNA secondary structure, including the kissing loop interactions, in the standard dot-bracket notation.

The pipeline, as shown in Figure 6a, consists of three modules. The first module, framed in the "Bitmap to grid graph" box in Figure 6a, is used to manually design a 2D pattern as a bitmap image and convert it to a grid graph representation in the standard DIMACS format. For the current example, the input and output of this module are shown in Figures 6b and 6c, respectively. Note that the edges and vertices bounding the selected pixels define the grid graph. Hence, the right vertical edge bounding the unselected pixel is not part of the output grid graph. The second module reads the input grid graph and searches, using Algorithm 1, for a minimum cost spanning tree. The output spanning tree for the running example is shown in Figure 6d. The module outputs the spanning tree in DIMACS format with additional comments on the starting position and direction of the minimum cost strand routing. The final module performs a twice around the tree traversal and generates a secondary structure in dot-bracket notation, augmented with information about the kissing-loop interactions. In our

example, the resulting secondary structure is shown in Figure 6e, where the matching square brackets indicate the kissing-loop pairings. This module also allows one to input secondary structure parameters and other design choices such as the number of turns per one horizontal edge, size and structure of the perimeter caps, kissing loop design, etc.

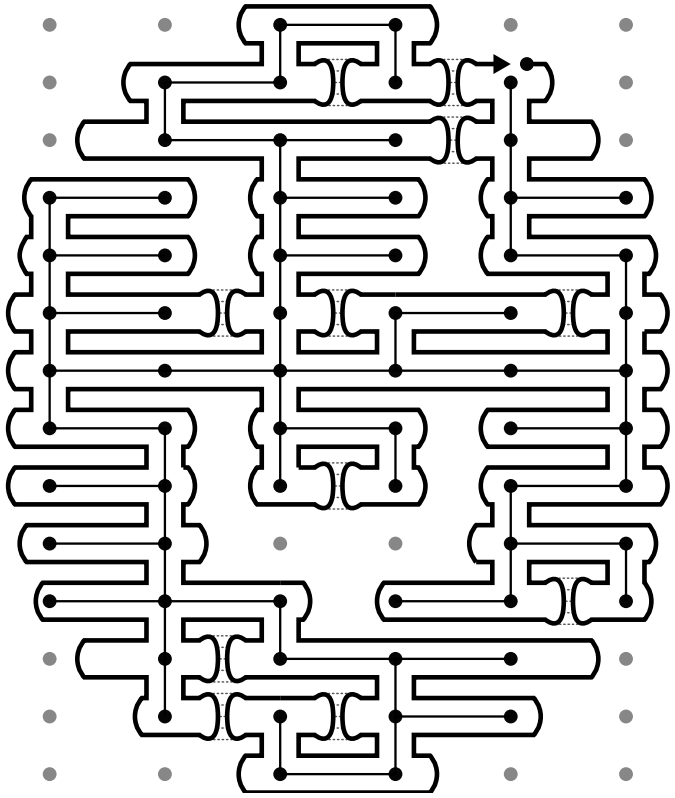


Fig. 7: A (near) zero cost strand routing of a smiley-face shape.

We demonstrate the capability of our software pipeline to produce polymerase-untrapped designs by running it on our grid representation of the smiley-face shape from [6]. We carved the smiley-face from a 14 by 6 canvas by deleting pixels corresponding to the two eyes, the mouth and the background. Our algorithm produced a spanning tree and routing, as shown in Figure 7, which only has the trivial cost. In the figure, black vertices correspond to the input grid

graph, while grey vertices are only placed to hint at the canvas from which the shape was carved. The fact that our pipeline found a (near) zero cost solution illustrates the utility of our algorithmic approach for finding designs that avoid polymerase trapping even in relatively large shapes. Nevertheless, we note, for instance, that our supplementary two-crossover-per-row constraint is insufficient to overcome flexibility in partial grids and further modelling is required to fully capture other constraints of RNA design of complex shapes.

6 Conclusions and Future Work

In the framework of RNA origami tile design, we have identified the topological folding obstacle of polymerase trapping, formulated it as a combinatorial problem, and designed an optimisation procedure and operational software to minimise the risk of encountering this obstacle. The software pipeline still needs to be extended to include sequence generation, but this involves several further considerations that we are currently investigating.

In the process, we have observed that in fact zero-cost routings (according to our present cost measure) are quite prevalent, and are planning another paper on a combinatorial characterisation of those.

In the actual biochemical setting, our present cost measure and design constraints are certainly too simplistic, and other considerations need to be taken into account. However the optimisation framework should be able to accommodate such changes quite conveniently.

Acknowledgments. We thank Ebbe Andersen and Cody Geary for introducing us to the problem of polymerase trapping in RNA origami tile design, and their encouragement to proceed with the solution approach discussed in this paper.

References

1. Chworos, A., Severcan, I., Koyfman, A.Y., Weinkam, P., Oroudjev, E., Hansma, H.G., Jaeger, L.: Building programmable jigsaw puzzles with RNA. Science 306(5704), 2068–2072 (2004), https://doi.org/10.1126/science.1104686
2. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms. MIT Press (2009)
3. Geary, C., Chworos, A., Verzemnieks, E., Voss, N.R., Jaeger, L.: Composing RNA nanostructures from a syntax of RNA structural modules. Nano Lett. 17(11), 7095–7101 (2017), https://doi.org/10.1021/acs.nanolett.7b03842
4. Geary, C., Meunier, P.E., Schabanel, N., Seki, S.: Programming biomolecules than fold greedily during transcription. In: Proceedings, 41st Intl. Conf. on Mathematical Foundations of Computer Science (MFCS 2016), LIPIcs, vol. 58, pp. 43:1–43:14. Dagstuhl Publishing (2016), http://doi.org/10.4230/LIPIcs.MFCS.2016.43
5. Geary, C., Rothemund, P.W.K., Andersen, E.S.: A single-stranded architecture for cotranscriptional folding of RNA nanostructures. Science 345(6198), 799 (2014), https://doi.org/10.1126/science.1253920

6. Geary, C.W., Andersen, E.S.: Design principles for single-stranded RNA origami structures. In: Proceedings, 20th Intl. Conf. on DNA Computing and Molecular Programming (DNA20), LNCS, vol. 8727, pp. 1–19. Springer (2014), https://doi.org/10.1007/978-3-319-11295-4_1
7. Guo, P.: The emerging field of RNA nanotechnology. Nat. Nanotech. 5, 833–842 (2010), https://doi.org/10.1038/nnano.2010.231
8. Han, D., Qi, W., Myhrvold, C., Wang, B., Dai, M., Jiang, S., Bates, M., Liu, Y., An, B., F, Z., Yan, H., Yin, P.: Single-stranded DNA and RNA origami. Science 358(6369), eaao2648 (2017), https://doi.org/10.1126/science.aao2648
9. Jasinski, D., Haque, F., Binzel, D.W., Guo, P.: Advancement of the emerging field of RNA nanotechnology. ACS Nano 11(2), 1142–1164 (2017), https://doi.org/10.1021/acsnano.6b05737
10. Kohman, R., Kunjapur, A.M., Hysolli, E., Wang, Y., Church, G.M.: From designing the molecules of life to designing life: Future applications derived from advances in DNA technologies. Angew. Chem. Intl. Ed. accepted manuscript online (2018), https://doi.org/10.1002/anie.201707976
11. Kreweras, G.: Complexité et circuits eulériens dans les sommes tensorielles de graphes. Journal of Combinatorial Theory, Series B 24(2), 202–212 (1978), https://doi.org/10.1016/0095-8956(78)90021-7
12. Li, Y., Mao, C., Deng, Z.: Supramolecular wireframe DNA polyhedra: Assembly and applications. Chin. J. Chem. 35(6), 801–810 (2017), https://doi.org/10.1002/cjoc.201600789
13. Orponen, P.: Design methods for 3D wireframe DNA nanostructures. Natural Computing 17(1), 147–160 (2018), https://doi.org/10.1007/s11047-017-9647-9
14. Rothemund, P.W.K.: Folding DNA to create nanoscale shapes and patterns. Nature 440(7082), 297–302 (2006), https://doi.org/10.1038/nature04586
15. Seeman, N.C.: Structural DNA Nanotechnology. Cambridge University Press (2015)
16. Seeman, N.C., Sleiman, H.F.: DNA nanotechnology. Nature Reviews Materials 3, 17068 (2017), https://doi.org/10.1038/natrevmats.2017.68
17. Westhof, E., Masquida, B., Jaeger, L.: RNA tectonics: Towards RNA design. Folding and Design 1(4), R78–R88 (1996), https://doi.org/10.1016/S1359-0278(96)00037-5
18. Zhang, F., Nangreave, J., Liu, Y., Yan, H.: Structural DNA nanotechnology: State of the art and future perspective. J. Am. Chem. Soc. 136(32), 11198–11211 (2014), https://doi.org/10.1021/ja505101a