Automated rendering of multi-stranded DNA complexes with pseudoknots

Małgorzata Nowicka[1], Vinay K. Gautam[2], and Pekka Orponen[1]

[1] Department of Computer Science, Aalto University, Finland
malgorzata.nowicka@aalto.fi, pekka.orponen@aalto.fi
[2] Department of Chemical Engineering, NTNU, Norway
vinay.k.gautam@ntnu.no

Abstract. We present a general method for rendering representations of multi-stranded DNA complexes from textual descriptions into 2D diagrams. The complexes can be arbitrarily pseudoknotted, and if a planar rendering is possible, the method will determine one in time which is essentially linear in the size of the textual description. (That is, except for a final stochastic fine-tuning step.) If a planar rendering is not possible, the method will compute a visually pleasing approximate rendering in quadratic time. Examples of diagrams produced by the method are presented in the paper.

Keywords: DNA strand displacement, DSD systems, multi-strand DNA, pseudoknots, graph drawing, visualisation

1 Background

Multi-stranded DNA complexes are of wide interest in the area of DNA nanotechnology. In the present work, we focus specifically on their use as building blocks in DNA strand displacement (DSD) systems [40], which are a widely-used methodology for embedding computational processes in the interactions of pre-designed short DNA strands.

The design and modelling of DSD systems is typically done at the binding-domain level, where a *domain* is a contiguous and functionally distinct sequence of nucleotides. A DNA *strand* is then a sequence of connected domains. Strands are oriented, starting at the *5-end* and ending at the *3-end*. Domains of different strands may bind with each other in antiparallel orientation, forming a DNA complex which is (mathematically speaking) a multiset of strands connected by domain bindings. At the design phase one is however generally not interested in individual strands and complexes, but rather distinct similarity types (equivalence classes) of these. A similarity type of DNA complexes is called a DNA *species*, whereas with some abuse of terminology the term "strand" is retained also for similarity types of strands.

Automated tools for designing and modelling DSD systems commonly follow a four-stage pipeline (Figure 1) in which we are presently concerned with the final step of visualisation — or more specifically the task of producing 2D diagrams

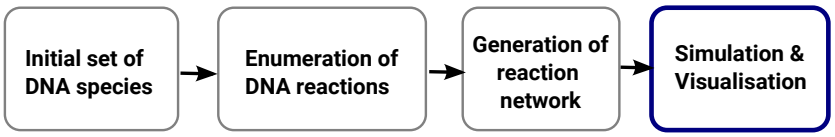

Fig. 1: DSD system design pipeline.

of species generated from an initial set of species. Such diagrams are highly useful to understand the characteristics of species emerging in a given DSD system, validate the system design, and communicate related results to a broader audience.

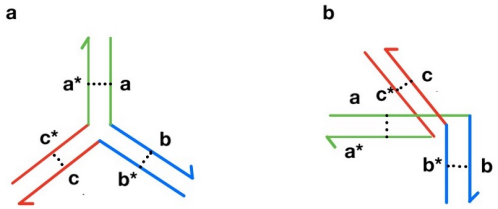

Fig. 2: DNA species diagrams. a: Good drawing. b: Unsatisfactory drawing.

As is customary in the DSD design literature [40], we aim to render multi-strand DNA species as 2D diagrams, where each strand is represented as a sequence of contiguous line or curve segments that depict the domains, and domain pairings are indicated by some connector symbol, for instance a line segment connecting the domain midpoints (Figure 2). Ideally, a 2D rendering of a given DNA species should satisfy the following conditions: (i) the domain depictions are straight-line segments of equal length; (ii) bound domains are drawn close to each other, ideally at a fixed distance and in antiparallel orientation; (iii) the polylines (or polycurves) representing species do not cross each other in the drawing plane. For structurally complex species, the task of satisfying all these constraints simultaneously and satisfactorily can be quite difficult or even impossible.

The challenge of pseudoknots Many current DNA species rendering algorithms take inspiration from methods used in visualising RNA secondary structures. In RNA, a single strand folds upon itself and forms bindings pairing its nucleotides. In simple "nested" pairing arrangements, this process results in a tree-like secondary structure with contiguous *stem* segments of paired bases constituting the edges and *loops* of unpaired bases forming the leaves (Figure 3). Visualisation methods for such structures commonly follow this intrinsic tree layout [6,31].

This approach extends to DNA visualisations as well, because if the strands constituting a multi-strand DNA species are arranged in some linear order and

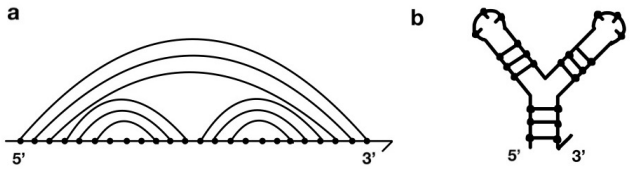

Fig. 3: RNA secondary structure diagrams. a: A 1D arc diagram depicting base pairings. b: Corresponding 2D secondary structure diagram.

connected into a single strand, the rendering problem reduces to the corresponding one for RNA secondary structures. Thus, the only additional task is to determine a good ordering for the strands. (Of course for n strands there are $n!$ possible orderings, so the task is not quite straightforward.) This method is widely used in visualisations of nested DNA species, for instance in the DyNAMiC Workbench tool [16].

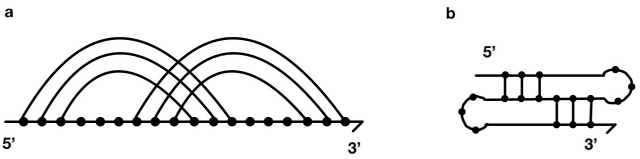

Fig. 4: An H-type RNA pseudoknot. a: Arc diagram. b: Secondary structure diagram.

A problem emerges, however, when base pairings are not nested, but cross each other in the linear strand arrangement, as in Figure 4a. In this *pseudoknotted* situation, the secondary structure no longer has a tree-like character (Figure 4b), and many challenges ensue both in diagramming and otherwise analysing the structure [1,25].

A number of RNA visualisers nevertheless support the rendering of pseudoknotted structures using alternative approaches. PseudoViewer3 [7] identifies and draws known types of pseudoknots, and arranges the rest of the structure in relation to them. The tool jViz.Rna 2.0 [36] uses a force-directed algorithm for assembling the nucleotides, whereas jViz.RNA 4.0 [30] and forna [23] identify the pseudoknotting connections and draw them on top of a pruned pseudoknot-free skeleton of the species.

A multi-strand DNA species can be considered pseudoknotted when no linear ordering of its constituent strands results in a nested base-pairing arrangement. Rendering of pseudoknotted species in DSD systems has been little studied, because when there is no tree structure to be identified, exploring the space of $n!$ permutations for a species of n strands becomes challenging. VisualDSD [24], a widely-used tool for designing and modelling DSD systems, takes the approach

of rendering a pseudoknot-free skeleton of the species and then just highlighting the pseudoknot-inducing domains, without connecting them visually. While in RNA this approach may be helpful in order to easily identify the type of the pseudoknot, in DNA it is less useful in illustrating the structure. (For examples, see Figure 7.) A force-directed approach was used in [15] but it covers only relatively simple pseudoknotted structures. Other common DSD system design tools such as NUPACK [39] and DyNAMiC Workbench [16] do not provide options for visualising pseudoknots.

2 A DNA species rendering algorithm

We approach the task of rendering multi-strand DNA species in a Gordian-knot manner: since a key target is to achieve a noncrossing arrangement for the strands, we aim to achieve this directly by a planar graph embedding technique that is guaranteed to find such an arrangement efficiently if one exists. For cases where no planar (noncrossing) representation is possible, we present an efficient and visually appealing approximate method.

Luckily, quite many pseudoknotted (RNA and multi-strand DNA) structures have planar renderings: a simple example is the H-type pseudoknot in Figure 4, for which a planar drawing is presented in subfigure 4b. In RNA, planar pseudoknots are closely related to the Rivas-Eddy class of secondary structures, which is also one of the largest known algorithmically tractable families of RNA structures [13,26,29].

Text representation For textual description of multi-strand DNA species we use a 'process calculus' notation based on named-domain pairings and introduced to the DSD context in [27] (see Figure 5a). Lowercase letters and numbers are used as domain names (e.g. a). A complementary domain has the same name with an additional asterisk appended (e.g. a^*). A domain with a given name can appear repeatedly. A pairing of domains is indicated by an exclamation mark followed by a matching index, appended to the domain names (e.g. $a!1$, $a*!1$). Each pairing must be unique. A strand comprises one or more sequentially connected domains, enclosed in angle brackets: $\langle a\ b\rangle$. A species is a collection of strands, separated by two forward slashes: $\langle a\ b\rangle//\langle c\rangle$. Paired domains must belong to the same species in the description.

Graph representation and algorithm generality Figure 5a presents an example of a textual description of a species S and Figure 5b its graph representation G. The vertices in G correspond to the domain boundaries, including the 5- and 3-ends, of the strands in S. The domains within the strands (solid lines in Figure 5b) and the domain pairings connecting the strands (dashed lines in Figure 5b) constitute the edges in G. To ensure the antiparallel orientation of each domain pairing $\{a, a^*\}$, the 5-end of a is connected to the 3-end of a^* and vice versa. We say that a species S is *planar* if G is a planar graph, i.e. can be embedded on a plane without edge crossings. Determining if S is planar can be done in

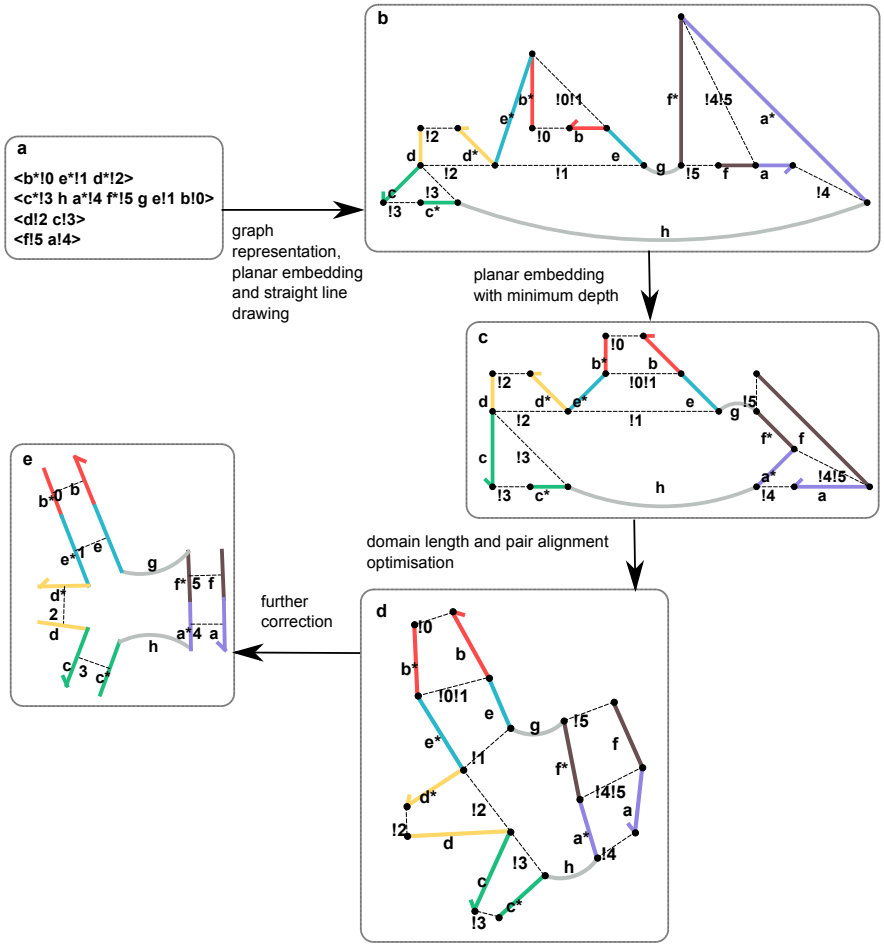

Fig. 5: Rendering steps of a species S. Each box displays a stage of the rendering algorithm. The captions on the arrows indicate the operations taken to achieve the next step. a: Text representation. b: Straight-line drawing. Vertices mark the ends of domains, solid lines represent the domains and dashed lines the pair connections. c: Minimised depth. d: Initial optimisation. e: Final drawing.

linear time by using any planarity testing algorithm, e.g. the Boyer-Myrvold edge addition method [5].

To characterise the generality of the algorithm we consider the notion of *book thickness* of a graph [2] that is widely used in the context of RNA theory [21]. For a given graph G a *book embedding* comprises a linear ordering of the vertices of G, called the book *spine*, and a partition of its edges. The edges in each class of the partition are drawn as non-crossing arcs on a separate halfplane (a *page*) bordered on the book spine. The book thickness of G is then the minimum number of pages needed to achieve such a book embedding, for some ordering of the vertices along the spine. It is known that the book thickness of a graph G is at most 2 if and only if G is a subgraph of a planar Hamiltonian graph [2].

In RNA the vertices represent nucleotides and the edges their pairings [21]. There is only one possible ordering of the vertices, and in case of linear RNA, connecting the first and the last vertex makes the graph Hamiltonian. Therefore the existence of a planar rendering of an RNA secondary structure is equivalent to the condition that its arc diagram can be embedded on at most 2 pages. This can be tested by e.g. checking the bipartiteness of an inconsistency graph $\theta(V, E)$ [21], where every vertex corresponds to a crossing of the arcs in the arc diagram.

In multi-strand DNA, the ordering of the vertices is not given, which makes the problem of determining if a 2-page embedding exists NP-complete [10,37]. Moreover, there exist maximal planar graphs that are not Hamiltonian, therefore some planar graphs might need more than 2 pages (see Figure 7c). In fact, any planar graph can be embedded on 4 pages [22,38]. Therefore the notion of planar DNA species as defined in this paper encompasses all 1-page (= nested) species, all 2-page pseudoknots, and some pseudoknots that require more than 2 pages and are planar.

Straight-line drawing It is known that any planar simple graph (= no multi-edges, no self-loops) has a plane embedding with straight-line edges [35,12,32]. By definition, a species graph G as described earlier is always simple, and can thus be drawn with straight lines if it represents a planar species. Rendering G this way satisfies the conditions that the domains are drawn as straight-line segments and the shortest distance between domains within a pair is represented by a straight line.

Given a planar species graph G, a drawing (plane embedding) D of G can be achieved in linear time by an algorithm described in [9] that produces crossing-free, straight-line drawings on an $(n-2) \times (n-2)$ grid for graphs with n vertices. The algorithm comprises four stages:

1. G is augmented by additional edges to achieve the biconnectivity required for the canonical decomposition [20] applied in step 3.
2. A combinatorial embedding of G (clockwise ordering of incident edges at each vertex) is determined e.g. with the Boyer-Myrvold method [5]. The face of G with most vertices is chosen as the external face.

3. The canonical decomposition is computed. This is an ordered partition of the set of vertices of G and determines the order of the vertices in the process of drawing.
4. The graph is drawn on a plane according to the algorithm in [9]. The augmented edges are removed from D.

All stages of the algorithm are of linear time complexity. An example result is presented in Figure 5b.

The aforementioned algorithm covers only planar graphs, and a different approach needs to be adopted for the non-planar cases. Our non-planar drawing method follows the topology-shape-metrics approach [11,34] for orthogonal drawings, where the edges may be either straight or bend 90 degrees. It prioritises the minimisation of the number of crossings and adheres to other aesthetic criteria such as minimising the number of edge bends, edge length, and the area of the drawing. These criteria correspond to our criteria of a good DNA species drawing with no crossings and close distances between the paired domains. The algorithm comprises three stages:

1. G is planarised by first finding some maximal planar subgraph of G and then reinserting the remaining edges in a way that minimises the number of crossings [18]. A dummy vertex is created at the location of each crossing. From here, a planar embedding is computed using the same method as in step 2 of the planar drawing.
2. Given the planar embedding, the orthogonalisation stage of the algorithm constructs a network in which the minimum number of bends in G is determined by a min-cost flow method described in [33].
3. Lastly, the compaction stage determines the lengths of the edges and coordinates of the vertices, producing a drawing of minimum possible area. In the final rendering of D, the dummy vertices are removed, edge bending is ignored and the edges are represented with straight lines. This may lead to crossings in D.

The time complexity of this algorithm ranges from $O(n)$ to $O((n+c)^2 \log n)$, where c is the number of crossings, depending on the planarity of the graph and the orthogonalisation and compaction methods [34].

Minimum depth and maximum external face drawing The *depth* of a graph is a measure of the topological nesting of its biconnected components (maximal biconnected subgraphs) [28]. In a species graph G, the biconnected components comprise connected domains and domain pairs. When depth is not minimised, some domains are topologically nested inside other pairs, which prevents the outer pair from achieving the desired proximity between its domains.

Consider a planar embedding of a graph of a species with strands $\langle e!1\ f\rangle\ \langle e*!1\rangle$. Let r be a vertex between domains e and f and $!1$ be the edge connecting the bound domains. There exist two possible combinatorial embeddings in r (see Figure 6d) but only the one of minimum depth will result in a drawing that minimises the distance between the paired domains. In Figure 5b,

pairs aa^* bb^*, and ff^* are topologically nested, and therefore pairs cc^*, dd^*, and ee^* cannot properly align.

In order to minimise the depth, a linear time embedding algorithm [19] that uses an SPQR-tree data structure is applied to the second stage of the planar G drawing. It chooses a planar embedding of maximum external face size among all planar embeddings with minimum depth. Embeddings with maximum external face have shown to improve certain metrics like the length of the edges and the size of the area [17]. The result of the new embedding is presented in Figure 5c.

Drawing optimisation The final step for both planar and non-planar G is to adjust the lengths of the edges without introducing new crossings in D. This consideration imposes an additional requirement on force-directed graph layout methods commonly used for such purposes. In a force-directed layout algorithm, the connected vertices are attracting each other until the distance between them is of desirable length, and the unconnected vertices are repelling each other. An additional node-edge repulsion formula tries to prevent the overlapping of nodes and edges, but due to the discreteness of the movements of the vertices, new crossings may appear. Article [3] introduces the use of zones around every vertex that confine its movement at each iteration of the algorithm and prevent the formation of new crossings (see Figure 5d).

Further correction in the placement of the domains is required, as the previous step does not achieve the desired lengths for all the edges or the right alignment between domains in a pair. A simulated annealing algorithm [4] is used. Let domains d_1 and d_2 be paired and represented by vectors, such that $d_1 = [d_1^3, d_1^5]$ and $d_2 = [d_2^3, d_2^5]$. A distance to the correct placement of d_1 and d_2 is computed in three steps as follows (see Figure 6).

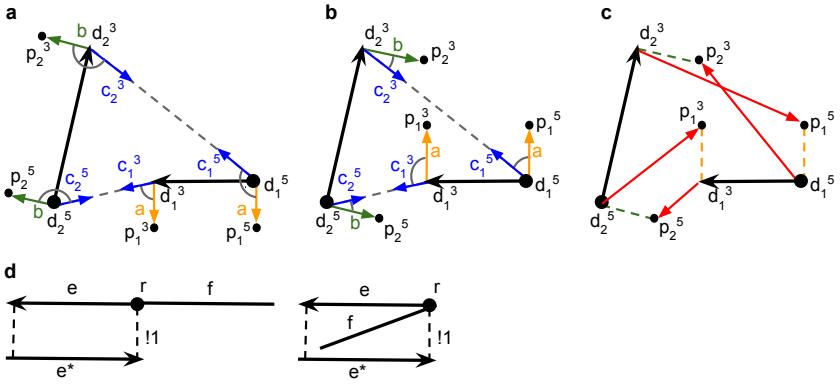

Fig. 6: Computing the correct placement of d_1 and d_2 and choosing the correct embedding for the vertex r.

1. Let a and b be vectors rotated 90 degrees counter-clockwise in relation to d_1 and d_2 respectively. Let p_1^3 and p_1^5 be the endpoints of a anchored at both ends of d_1 and p_2^3 and p_2^5 be the endpoints of b anchored at both ends of d_2. Let $c_1^3 = [d_1^3, d_2^5]$, $c_1^5 = [d_1^5, d_2^3]$, $c_2^3 = [d_2^3, d_1^5]$, and $c_2^5 = [d_2^5, d_1^3]$ (Figure 6a).
2. The correct side for d_1 to bind with d_2 is determined by the inequality:

$$\text{sign}(\cos(a, c_1^3)) + \text{sign}(\cos(a, c_1^5)) + \text{sign}(\cos(b, c_2^3)) + \text{sign}(\cos(b, c_2^5)) \geq 3$$

   where $\text{sign}(x) = 0$ if $x \leq 0$, and 1 otherwise. If the inequality is not satisfied, rotate a and b so that they are in 90 degrees clockwise relation to d_1 and d_2 respectively, as shown in Figure 6b. Note the change of the position of the endpoints of a and b.
3. Compute the distances from d_2^5 to p_1^3, d_2^3 to p_1^5, d_1^5 to p_2^3, and d_1^3 to p_2^5 as in Figure 6c.

At each step of the algorithm, three factors are being optimised: the distance to the correct placement of domains for all pairs, the deviation from the desired lengths of the edges, the deviation from the desired distance between the ends of loop domains.

Figure 5e presents the final outcome where, to follow standard practice, the double vertices at the domain boundaries have been removed and replaced by single edges connecting the respective paired domains.

Implementation The planar embeddings of minimum depth and maximum external face for both planar and non-planar species and the graph drawing were achieved with the help of algorithms in the Open Graph Drawing Framework [8]. The visualisation is implemented as a part of the XDSD design tool [14].

3 Results

Figure 7 demonstrates the visualisation possibilities of the algorithm. The results are compared with VisualDSD [24], a widely-used tool that accepts pseudoknotted multi-strand designs as inputs.

The rendering algorithm in VisualDSD admits multi-strand, pseudoknotted species with loops as inputs. The bonds that make the species a pseudoknot (pseudoknotting connections) are accepted but not specifically handled except for denoting them by boldening the domain and bond name.

Figure 7a presents a 1-page, nested species. The VisualDSD method does not maintain the connectedness of the strands and instead draws dashed lines between the separated domains. Our algorithm resolves to manipulating the lengths of the domains in order to keep the domains together.

The species in Figure 7b is a 2-page pseudoknot. The VisualDSD version breaks the species apart and superposes the pseudoknotting connections after the other parts of the species are rendered. Our method does not distinguish between the regular and pseudoknotting connections and renders all of them.

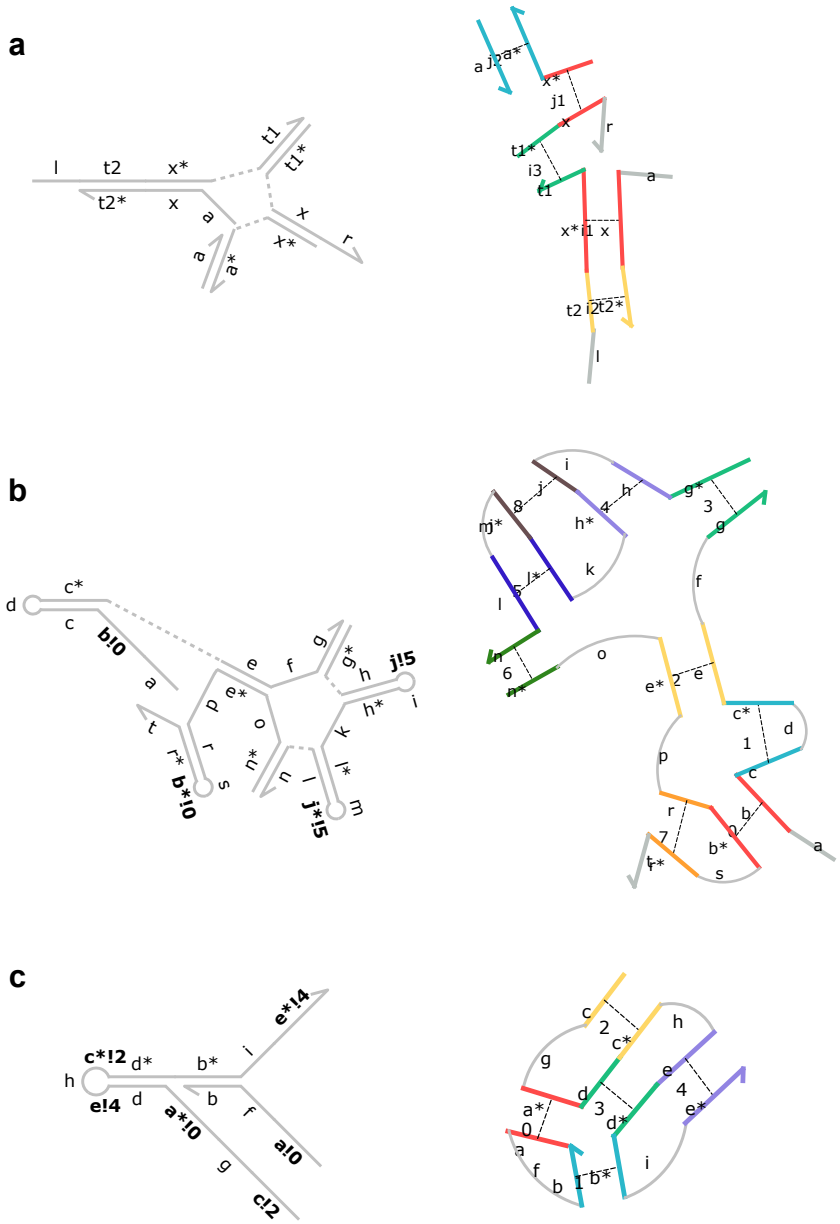

Fig. 7: Three species renderings generated by VisualDSD on the left and our algorithm on the right.

Figure 7c shows another pseudoknot, a 3-page species that has a planar representation. Again, the VisualDSD rendering ignores the pseudoknotting connections. These noticeably change the appearance of the species, once the pseudoknotting domains are brought together.

4 Conclusions

Discussion Based on notions of planar pseudoknots and planar multi-strand DNA species, we have presented a rendering method that is able to closely follow the conventions of DSD system drawings (strands do not cross each other, domains are represented as fixed length straight-line segments, and bound domains are close to each other at a fixed distance) while keeping the strands connected and pseudoknots visualised.

In renderings for pseudoknot-free species, conventionally there is one binding side for all the domains in the species (e.g. clockwise or to the right w.r.t. the 5-3 direction of the domains). Our algorithm on the other hand does not distinguish between pseudoknotted and pseudoknot-free species, and automates the embedding of the species without paying attention to the consistency of the binding sides (see Figure 6a). This may in some cases lead to decreased readability of the drawing.

Further work The current implementation of the embedder cannot be applied before the augmentation stage of the drawing algorithm. Therefore the augmented edges are embedded as well, which after their removal may result in a species rendering with crossings and non-minimal depth. In a future version, one could aim to apply the embedder before the graph is augmented, so that the addition and removal of the support edges does not affect the depth of the resulting graph.

The drawing optimisation now uses a simulated annealing approach with fixed hyperparameters that does not take into account the production of new crossings. This way the optimisation process is faster, but may exert too much or too little pressure on the species, resulting in new crossings or unaligned pairs of uneven-length domains respectively. Further developments of the drawing optimisation part are needed since this is the only non-linear step in the algorithm.

References

1. Akutsu, T.: Dynamic programming algorithms for RNA secondary structure prediction with pseudoknots. Discrete Applied Mathematics **104**, 45–62 (2000)
2. Bernhart, F., Kainen, P.C.: The book thickness of a graph. Journal of Combinatorial Theory, Series B **27**, 320–331 (1979)
3. Bertault, F.: A force-directed algorithm that preserves edge-crossing properties. Information Processing Letters **74**, 7–13 (2000)
4. Bertsimas, D., Tsitsiklis, J.: Simulated annealing. Statistical Science **8**, 10–15 (1993)

5. Boyer, J.M., Myrvold, W.J.: On the cutting edge: Simplified O(n) planarity by edge addition. Journal of Graph Algorithms and Applications **8**, 241–273 (2004)
6. Bruccoleri, R.E., Heinrich, G.: An improved algorithm for nucleic acid secondary structure display. Comput. Appl. Biosci. **4**, 167–173 (1988)
7. Byun, Y., Han, K.: PseudoViewer3: Generating planar drawings of large-scale RNA structures with pseudoknots. Bioinformatics **25**, 1435–1437 (2009)
8. Chimani, M., Gutwenger, C., Jnger, M., Klau, G.W., Klein, K., Mutzel, P.: The Open Graph Drawing Framework (OGDF). In: Handbook of Graph Drawing and Visualization, chap. 17. CRC Press (2014)
9. Chrobak, M., Kant, G.: Convex grid drawings of 3-connected planar graphs. International Journal of Computational Geometry & Applications **7**, 211–223 (1997)
10. Chung, F.R.K., Leighton, F.T., Rosenberg, A.L.: Embedding graphs in books: A graph layout problem with applications to VLSI design. SIAM J. Algebraic Discrete Methods **8**, 33–58 (1986)
11. Di Battista, G., Eades, P., Tammasia, R., Tollis, I.G.: Graph Drawing. Prentice Hall, Upper Saddle River, NJ, United States (1999)
12. Fáry, I.: On straight-line representation of planar graphs. Acta Sci. Math. **11**, 229–233 (1948)
13. Gao, S., Ding, K.: A graphical criterion of planarity for RNA secondary structures with pseudoknots in Rivas–Eddy class. Theoretical computer science **395**, 47–56 (2008)
14. Gautam, V.K., Long, S., Nowicka, M., Orponen, P.: XDSD: A Tool Bridging DSD System Design to Rule-Based Modelling and Simulation. Web server: `https://xdsd-web.org`
15. Gautam, V.K., Long, S., Orponen, P.: RuleDSD: A rule-based modelling and simulation tool for DNA strand displacement systems. Proceedings of the 13th International Joint Conference on Biomedical Engineering Systems and Technologies pp. 158–167 (2020)
16. Grun, C., Werfel, J., Zhang, D.Y., Yin, P.: DyNAMiC Workbench: An integrated development environment for dynamic DNA nanotechnology. J. R. Soc. Interface **12**(111), 20150580 (2015)
17. Gutwenger, C.: Application of SPQR-trees in the planarization approach for drawing graphs. Ph.D. thesis, Technische Universität Dortmund (2010)
18. Gutwenger, C., Mutzel, P.: An experimental study of crossing minimization heuristics. In: Graph Drawing: 11th International Symposium 2003. pp. 13–24. LNCS 2912 (2004)
19. Gutwenger, C., Mutzel, P.: Graph embedding with minimum depth and maximum external face. In: Graph Drawing: 11th International Symposium 2003. pp. 259–272. LNCS 2912 (2004)
20. Harel, D., Meir, S.: An algorithm for straight-line drawing of planar graphs. Algorithmica **20**, 119–135 (2000)
21. Haslinger, C., Stadler, P.F.: RNA structures with pseudo-knots: Graph-theoretical, combinatorial, and statistical properties. Bulletin of Mathematical Biology **61**, 437–467 (1999)
22. Kaufmann, M., Bekos, M., Klute, F., Pupyrev, S., Raftopoulou, C., Ueckerdt, T.: Four pages are indeed necessary for planar graphs. Journal of Computational Geometry **11**, 332–353 (2020)
23. Kerpedjiev, P., Hammer, S., Hofacker, I.L.: Forna (force-directed RNA): Simple and effective online RNA secondary structure diagrams. Bioinformatics **31**, 3377–3379 (2015)

24. Lakin, M.R., Youssef, S., Polo, F., Emmott, S., Phillips, A.: Visual DSD: a design and analysis tool for DNA strand displacement systems. Bioinformatics **27**, 3211–3213 (2011)
25. Lyngsø, R.B.: Complexity of pseudoknot prediction in simple models. In: Automata, Languages and Programming. ICALP 2004. pp. 919–931. LNCS 3142 (2004)
26. Lyngsø, R.B., Pedersen, C.N.S.: RNA pseudoknot prediction in energy-based models. Journal of Computational Biology **7**, 409–427 (2000)
27. Petersen, R.L., Lakin, M.R., Phillips, A.: A strand graph semantics for DNA-based computation. Theoretical Computer Science **632**, 43–73 (2016)
28. Pizzonia, M., Tamassia, R.: Minimum depth graph embedding. Algorithms - ESA 2000, 8th Annual European Symposium **ESA 2000, LNCS 1879**, 356–267 (2000)
29. Rivas, E., Eddy, S.R.: A dynamic programming algorithm for RNA structure prediction including pseudoknots. Journal of Molecular Biology **285**, 2053–2068 (1999)
30. Shabash, B., Wiese, K.C.: jViz.RNA 4.0 visualizing pseudoknots and rna editing employing compressed tree graphs. PLoS ONE **14**(5), e0210281 (2019)
31. Shapiro, B.A., Maizel, J., Lipkin, L.E., Currey, K., Whitney, C.: Generating non-overlapping displays of nucleic acid secondary structure. Nucleic Acids Res. **12**, 75–88 (1984)
32. Stein, S.K.: Convex maps. Proceedings of the American Mathematical Society **2**, 464–466 (1951)
33. Tamassia, R.: On embedding a graph in the grid with the minimum number of bends. SIAM Journal on Computing **16**, 421–444 (1987)
34. Tamassia, R., Di Battista, G., Batini, C.: Automatic graph drawing and readability of diagrams. IEEE Trans. Syst. Man Cybern. **18**, 61–79 (1988)
35. Wagner, K.: Bemerkungen zum Vierfarbenproblem. Jahresbericht der Deutschen Mathematiker-Vereinigung **46**, 26–32 (1936)
36. Wiese, K.C., Glen, E., Vasudevan, A.: jViz.Rna a Java tool for RNA secondary structure visualization. IEEE Trans. NanoBioscience **4**, 212–218 (2005)
37. Wigderson, A.: The complexity of the Hamiltonian circuit problem for planar graphs. Tech. Rep. 298, Princeton University (1982)
38. Yannakakis, M.: Planar graphs that need four pages. Journal of Combinatorial Theory, Series B **145**, 241–263 (2020)
39. Zadeh, J., Steenberg, C., Bois, J., Wolfe, B., Pierce, M., Khan, A., Dirks, R., Pierce, N.: NUPACK: Analysis and design of nucleic acid systems. Journal of Computational Chemistry **32**, 170–3 (2011)
40. Zhang, D.Y., G., S.: Dynamic DNA nanotechnology using strand-displacement reactions. Nature Chemistry (2011)