Design and modular self-assembly of nanostructures

Joakim Bohlin

Balliol College
University of Oxford

A thesis submitted for the degree of Doctor of Philosophy

Michaelmas 2021

Abstract

Your abstract text goes here. Check your departmental regulations, but generally this should be less than 300 words. See the beginning of Chapter **??** for more.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque sit amet nibh volutpat, scelerisque nibh a, vehicula neque. Integer placerat nulla massa, et vestibulum velit dignissim id. Ut eget nisi elementum, consectetur nibh in, condimentum velit. Quisque sodales dui ut tempus mattis. Duis malesuada arcu at ligula egestas egestas. Phasellus interdum odio at sapien fringilla scelerisque. Mauris sagittis eleifend sapien, sit amet laoreet felis mollis quis. Pellentesque dui ante, finibus eget blandit sit amet, tincidunt eu neque. Vivamus rutrum dapibus ligula, ut imperdiet lectus tincidunt ac. Pellentesque ac lorem sed diam egestas lobortis.

Suspendisse leo purus, efficitur mattis urna a, maximus molestie nisl. Aenean porta semper tortor a vestibulum. Suspendisse viverra facilisis lorem, non pretium erat lacinia a. Vestibulum tempus, quam vitae placerat porta, magna risus euismod purus, in viverra lorem dui at metus. Sed ac sollicitudin nunc. In maximus ipsum nunc, placerat maximus tortor gravida varius. Suspendisse pretium, lorem at porttitor rhoncus, nulla urna condimentum tortor, sed suscipit nisi metus ac risus.

Aenean sit amet enim quis lorem tristique commodo vitae ut lorem. Duis vel tincidunt lacus. Sed massa velit, lacinia sed posuere vitae, malesuada vel ante. Praesent a rhoncus leo. Etiam sed rutrum enim. Pellentesque lobortis elementum augue, at suscipit justo malesuada at. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent rhoncus convallis ex. Etiam commodo nunc ex, non consequat diam consectetur ut. Pellentesque vitae est nec enim interdum dapibus. Donec dapibus purus ipsum, eget tincidunt ex gravida eget. Donec luctus nisi eu fringilla mollis. Donec eget lobortis diam.

Suspendisse finibus placerat dolor. Etiam ornare elementum ex ut vehicula. Donec accumsan mattis erat. Quisque cursus fringilla diam, eget placerat neque bibendum eu. Ut faucibus dui vitae dolor porta, at elementum ipsum semper. Sed ultrices dui non arcu pellentesque placerat. Etiam posuere malesuada turpis, nec malesuada tellus malesuada.

Design and modular self-assembly of nanostructures

Joakim Bohlin

Balliol College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Michaelmas 2021

Acknowledgements

Personal

This is where you thank your advisor, colleagues, and family and friends.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum feugiat et est at accumsan. Praesent sed elit mattis, congue mi sed, porta ipsum. In non ullamcorper lacus. Quisque volutpat tempus ligula ac ultricies. Nam sed erat feugiat, elementum dolor sed, elementum neque. Aliquam eu iaculis est, a sollicitudin augue. Cras id lorem vel purus posuere tempor. Proin tincidunt, sapien non dictum aliquam, ex odio ornare mauris, ultrices viverra nisi magna in lacus. Fusce aliquet molestie massa, ut fringilla purus rutrum consectetur. Nam non nunc tincidunt, rutrum dui sit amet, ornare nunc. Donec cursus tortor vel odio molestie dignissim. Vivamus id mi erat. Duis porttitor diam tempor rutrum porttitor. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed condimentum venenatis consectetur. Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Aenean sit amet lectus nec tellus viverra ultrices vitae commodo nunc. Mauris at maximus arcu. Aliquam varius congue orci et ultrices. In non ipsum vel est scelerisque efficitur in at augue. Nullam rhoncus orci velit. Duis ultricies accumsan feugiat. Etiam consectetur ornare velit et eleifend.

Suspendisse sed enim lacinia, pharetra neque ac, ultricies urna. Phasellus sit amet cursus purus. Quisque non odio libero. Etiam iaculis odio a ex volutpat, eget pulvinar augue mollis. Mauris nibh lorem, mollis quis semper quis, consequat nec metus. Etiam dolor mi, cursus a ipsum aliquam, eleifend venenatis ipsum. Maecenas tempus, nibh eget scelerisque feugiat, leo nibh lobortis diam, id laoreet purus dolor eu mauris. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Nulla eget tortor eu arcu sagittis euismod fermentum id neque. In sit amet justo ligula. Donec rutrum ex a aliquet egestas.

Institutional

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 765703

If you want to separate out your thanks for funding and institutional support, I don't think there's any rule against it. Of course, you could also just remove the subsections and do one big traditional acknowledgement section.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut luctus tempor ex at pretium. Sed varius, mauris at dapibus lobortis, elit purus tempor neque, facilisis sollicitudin felis nunc a urna. Morbi mattis ante non augue blandit pulvinar. Quisque nec euismod mauris. Nulla et tellus eu nibh auctor malesuada quis imperdiet quam. Sed eget tincidunt velit. Cras molestie sem ipsum, at faucibus quam mattis vel. Quisque vel placerat orci, id tempor urna. Vivamus mollis, neque in aliquam consequat, dui sem volutpat lorem, sit amet tempor ipsum felis eget ante. Integer lacinia nulla vitae felis vulputate, at tincidunt ligula maximus. Aenean venenatis dolor ante, euismod ultrices nibh mollis ac. Ut malesuada aliquam urna, ac interdum magna malesuada posuere.

Abstract

Your abstract text goes here. Check your departmental regulations, but generally this should be less than 300 words. See the beginning of Chapter **??** for more.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque sit amet nibh volutpat, scelerisque nibh a, vehicula neque. Integer placerat nulla massa, et vestibulum velit dignissim id. Ut eget nisi elementum, consectetur nibh in, condimentum velit. Quisque sodales dui ut tempus mattis. Duis malesuada arcu at ligula egestas egestas. Phasellus interdum odio at sapien fringilla scelerisque. Mauris sagittis eleifend sapien, sit amet laoreet felis mollis quis. Pellentesque dui ante, finibus eget blandit sit amet, tincidunt eu neque. Vivamus rutrum dapibus ligula, ut imperdiet lectus tincidunt ac. Pellentesque ac lorem sed diam egestas lobortis.

Suspendisse leo purus, efficitur mattis urna a, maximus molestie nisl. Aenean porta semper tortor a vestibulum. Suspendisse viverra facilisis lorem, non pretium erat lacinia a. Vestibulum tempus, quam vitae placerat porta, magna risus euismod purus, in viverra lorem dui at metus. Sed ac sollicitudin nunc. In maximus ipsum nunc, placerat maximus tortor gravida varius. Suspendisse pretium, lorem at porttitor rhoncus, nulla urna condimentum tortor, sed suscipit nisi metus ac risus.

Aenean sit amet enim quis lorem tristique commodo vitae ut lorem. Duis vel tincidunt lacus. Sed massa velit, lacinia sed posuere vitae, malesuada vel ante. Praesent a rhoncus leo. Etiam sed rutrum enim. Pellentesque lobortis elementum augue, at suscipit justo malesuada at. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent rhoncus convallis ex. Etiam commodo nunc ex, non consequat diam consectetur ut. Pellentesque vitae est nec enim interdum dapibus. Donec dapibus purus ipsum, eget tincidunt ex gravida eget. Donec luctus nisi eu fringilla mollis. Donec eget lobortis diam.

Suspendisse finibus placerat dolor. Etiam ornare elementum ex ut vehicula. Donec accumsan mattis erat. Quisque cursus fringilla diam, eget placerat neque bibendum eu. Ut faucibus dui vitae dolor porta, at elementum ipsum semper. Sed ultrices dui non arcu pellentesque placerat. Etiam posuere malesuada turpis, nec malesuada tellus malesuada.

Contents

List of Figures

List of Tables

Glossary

2D, 3D . . . . Two- or three-dimensional, referring in this thesis to spatial dimensions of a self-assembly structure.

DNA . . . . . Deoxyribonucleic acid.

RNA . . . . . Ribonucleic acid.

PDB . . . . . The Protein Data Bank. Used in this thesis to refer to the file format used to save atomic structures.

Species . . . . Used in this thesis to denote a type of cube allowed by the polycube rule.

Polycube . . . A structure consisting of multiple cubes connected by their sides.

Far out in the uncharted backwaters of the unfashionable end of the western spiral arm of the Galaxy lies a small unregarded yellow sun. Orbiting this at a distance of roughly ninety-two million miles is an utterly insignificant little blue green planet whose ape-descended life forms are so amazingly primitive that they still think digital watches are a pretty neat idea.

— D. Adams, The Hitchhiker's Guide to the Galaxy

1 Introduction

Contents

This chapter explains the background and motivation for my project and introduces the two sub-projects I have been working on for the last year.

1.1 Scope of the thesis

1.2 Background

How do you design and assemble something on the nanoscale? Imagine building something with lego bricks; you choose the building blocks you want and use your hands to attach them where you want them to be. However, for nanoscale objects, it is difficult to have that level of control. Instead, more success has been had by imitating nature and letting the building blocks assemble themselves.

1.2.1 DNA design

Deoxyribonucleic acid (DNA) is a string-like molecule used to encode the genes of living systems [1]. These strings are made up of units called *nucleotides*, consisting of the connecting sugar-phosphate backbone as well as one of four possible bases: *adenine* (A), *thymine* (T) *cytosine* (C), and *guanine* (G).

Through Watson-Crick base-pairing, "A" forms two hydrogen bonds with "T", while "G" forms three with "C", making DNA double-stranded. Each strand has a directionality, conventionally represented as going from the 3' to the 5' end of the strand, making the duplex anti-parallel.

The bases of the duplex are hydrophobic, while the sugar-phosphate backbone is hydrophilic, which means that the bases "hide" on the inside of the duplex to avoid contact with water molecules. However, the backbone distance is about 6 Å (0.6 nm), while the bases would need to be at a distance of 3.3 Å (the thickness of the base) to not leave any room for water [1]. To solve this, the DNA duplex forms a double helix structure with a radius of about 9 Å, placing everything at an energetically comfortable distance.

While a single double-helix is the most natural confirmation, it is possible for strands to branch into multiple junctions. For example, the Holliday junction is a junction between four double-helical arms, shown in Figure 1.1 in one of its possible configurations.

By designing sequences with complementary domains for the intended duplex regions, it is possible to create many different DNA motifs and structures [2].

A breakthrough in the field of structural DNA nanotechnology was the DNA origami technique [3] a now popular and proven method for creating larger irregular structures using DNA. The principle behind it, as illustrated in Figure 1.2, is to use short staple strands to fold one long viral scaffold strand into the desired structure.

Using design tools such as caDNAno[5], it has become relatively easy to design structures of any given form. However, the size of the origami is limited by the length of the scaffold, which one of the motivations for researchers to investigate modular approaches, as described in Section 2.1.

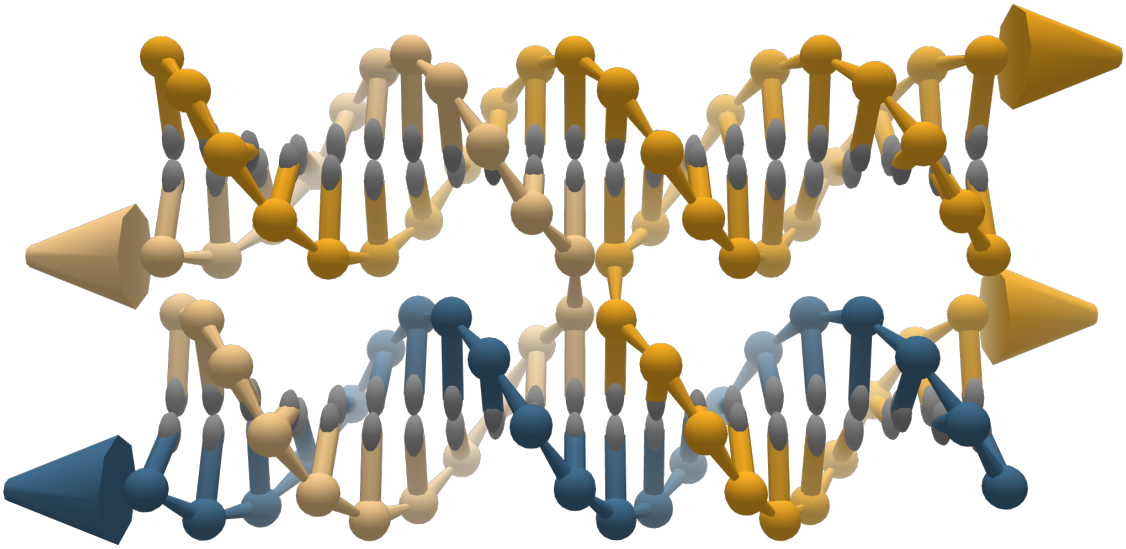

Figure 1.1: Holliday junction, designed in oxView. Cones at the end indicates the 3' end of each of the four strands.

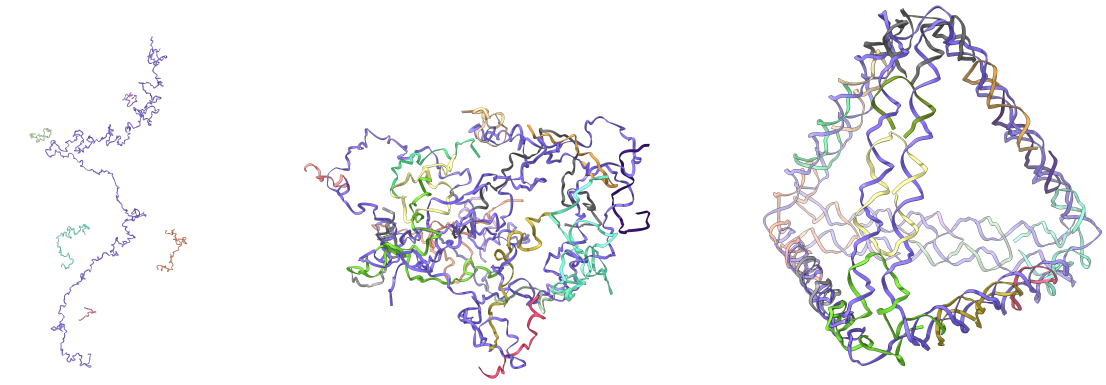

Figure 1.2: Illustration of DNA origami self-assembly of a tetrahedron. A long scaffold strand (purple), obtained from a virus, is folded into the desired shape by multiple short staple strands binding to complementary domains of the scaffold. Tetrahedron design obtained from `https://cando-dna-origami.org/examples/` and melted using oxDNA simulation [4].

1.2.2 RNA design

Ribonucleic acid (RNA) is very similar to DNA, but with the *thymine* base replaced by *uracil* (U). Just like DNA, it is possible to use it as a self-assembling building material.

Biologically, DNA is transcribed into RNA by the RNA polymerase enzyme as part of gene expression. While DNA folding is easier to predict, the fact that RNA is more reactive also offers the possibility of a more useful structure; for example

by incorporating aptamers, enzymes and other such functionalities[6].

Geary, et al., from the Andersen lab in Aarhus demonstrated a method[7–9] for co-transcriptionally folded RNA origami in 2014, which also enables folding *in vivo*. The design used a set of tertiary RNA motifs, such as kissing hairpins and double crossovers, to fold the transcribed RNA strand into the desired structure. The Andersen lab is one of the network partners, and I have spent a two-month secondment there working with their RNA origami method.

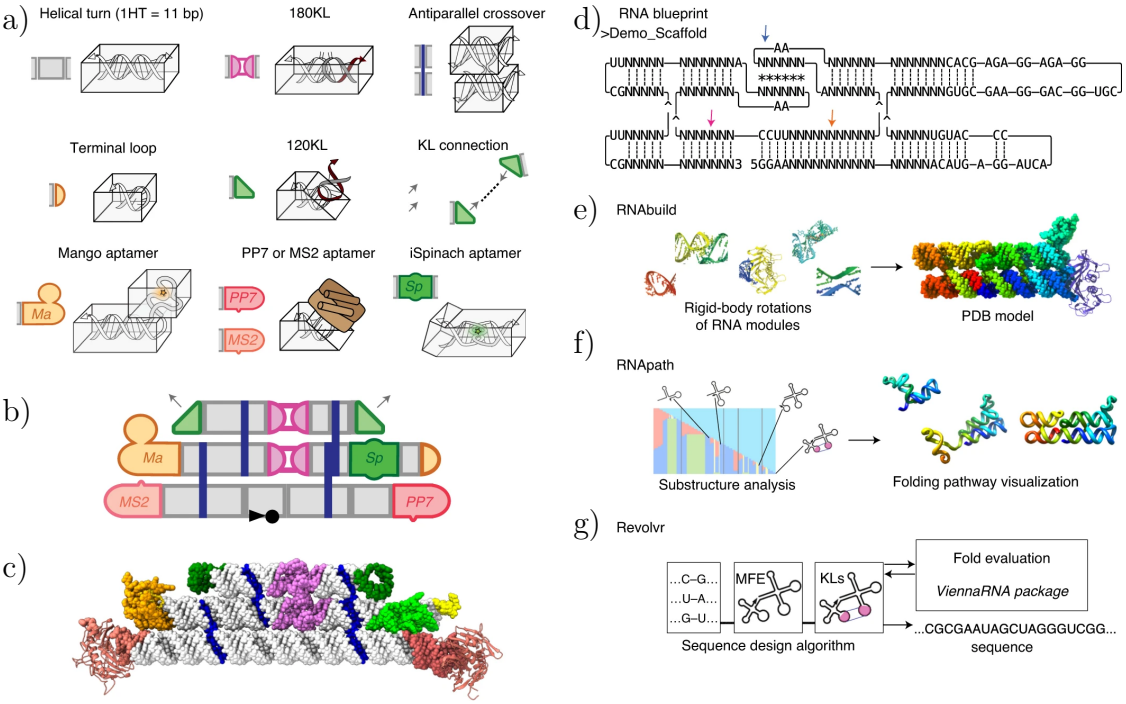

Figure 1.3: Co-transcriptional folding of RNA origami, adapted from [9]. a) The set of RNA motifs used as modular building blocks. b) Schematic of the modules connected to form a single strand. c) Atomistic model of the design in b). d) Shows the text-based blueprint used to create designs, while e), f), and g) shows scripts developed to aid the visualisation and preform sequence design for the origami.

Simplicity is prerequisite for reliability.

— Edsger W. Dijkstra

2

An introduction to modular assembly

Contents

From the start, the field of structural DNA nanotechnology has been interested in modular assemblies, although initially in the form of infinite crystal structures. Prof. Ned Seeman was inspired to pioneer the field after seeing the woodcut *Depth* by M.C. Escher [2], where fish are depicted organised into a neat crystalline structure.

However, components that self-assemble into chrystals can also be made to assemble into bounded structures, given that a way can be found to make them self-limiting. This chapter will provide an overview of the background of self-assembled

multicomponent structures, starting with experimental results of ever-increasing size and followed by a selection of theoretical models.

2.1 Experimental applications

From small tiles made from a handful of strands to megadalton-scale structures made from multiple origamis, modular self-assembly has seen considerable experimental research. This section will detail some results of particular interest to the polycube model presented in Chapter 3.

2.1.1 DNA tiles and bricks

Following early nucleic acid milti-arm junctions and lattices suggested by Seeman [10], Winfree [11] used double-crossover (DX) motifs, shown in Figure 2.2.a, to self-assemble 2D DNA crystals. As can be seen in figure 2.2.b, the lattices could be made with varying complexity; examplified using either two or for different species of tiles.

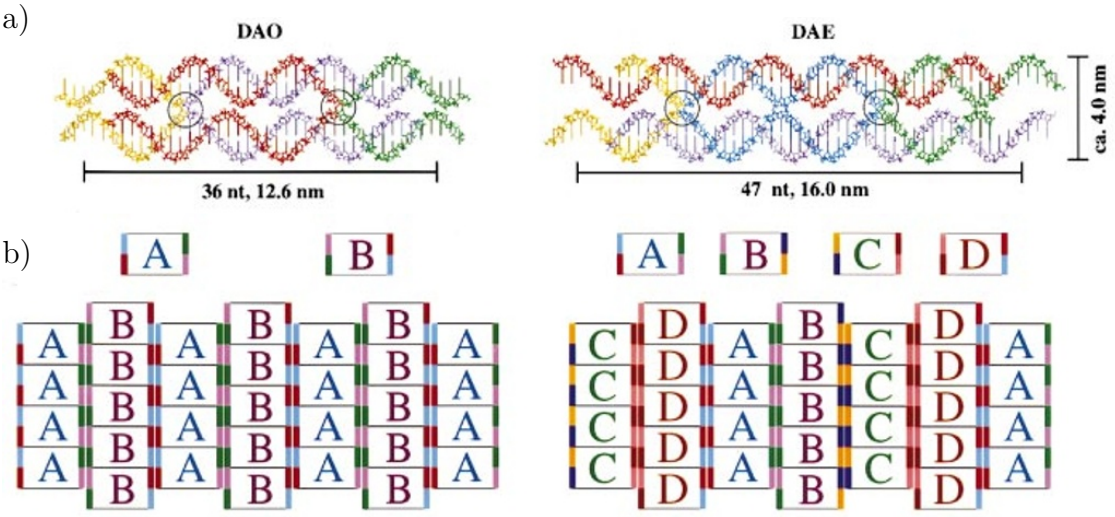

Figure 2.1: DX tiles forming 2D lattices, adapted from [11]. a) Examples of DAO and DAE tile designs. b) Lattices made from two and four tile species respectively.

2.1.2 RNA tiles

As covered in Section 1.2.2, it is possible to co-transcriptionally fold RNA origami [7]. This was shown in 2014, by folding RNA tiles that connect through complementary 120-degree kissing loop interactions, as seen in Figure 2.3.

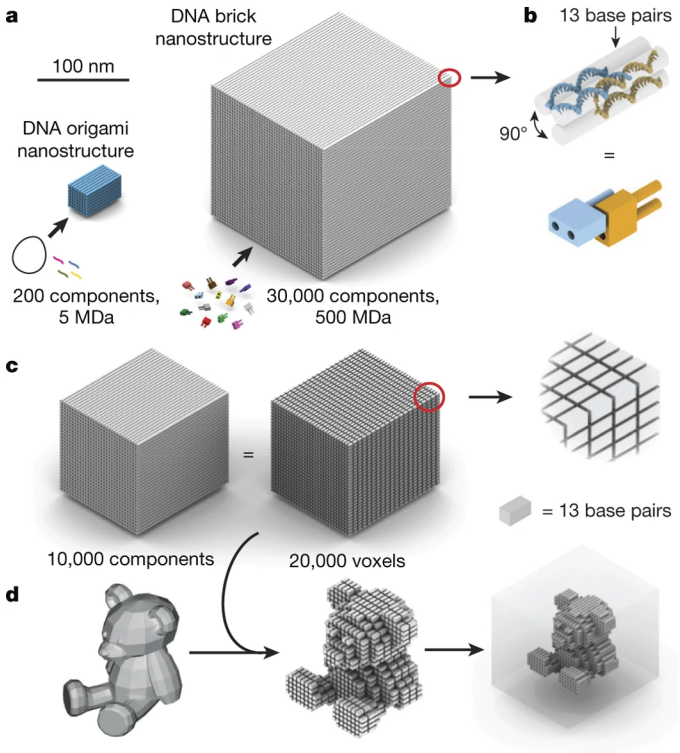

Figure 2.2: 3D bricks from [12]

2.1.3 DNA origami arrays

In 2017, Tikhomirov et al.[13] demonstrated two-dimensional patterns assembled on the micrometre-scale using square origami tiles connecting through their complementary edges. helices

2.1.4 Shape-complementary origami

Also, in 2017, Wagenbauer et al. [14] used shape-complementarity to assemble origami tiles into three-dimensional polyhedral shapes up to 450 nanometers in diameter.

[15]

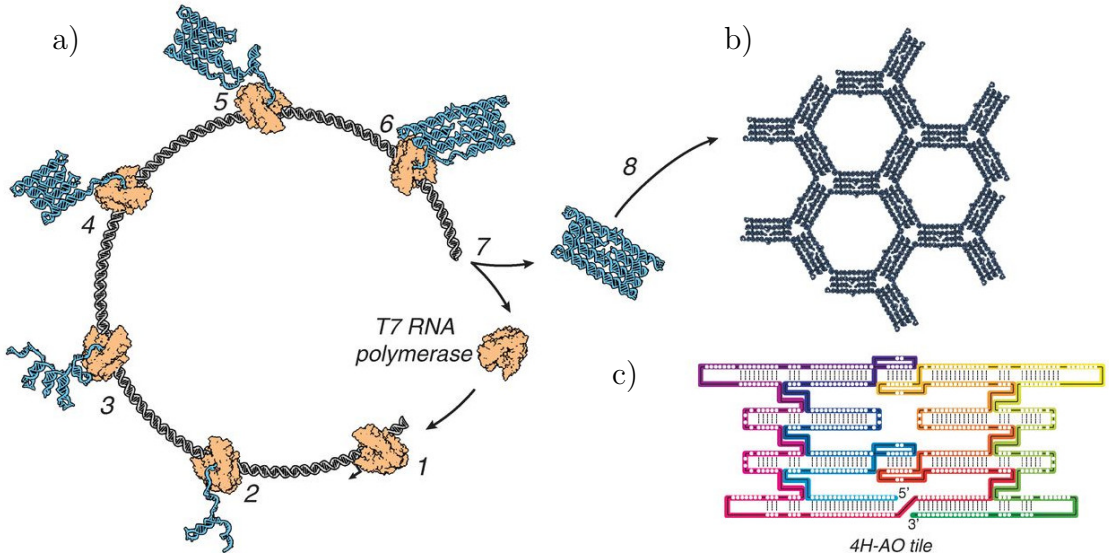

Figure 2.3: Co-transcriptional folding RNA origami tiles, adapted from [7]. The tiles connect through 120-degree kissing loop interactions, forming a hexagonal lattice. a) Co-transcriptional folding, where the RNA tile folds as it is transcribed from a DNA template. b) Hexagonal lattice formed by folded tiles. c) Detailed scematic of the four-helix 4H-AO tile, shown in a) and b).

2.1.5 DNA origami nanochambers

2.1.6 Octahedral DNA origami frames

2.2 Modular assembly models

It would be computationally unreasonable to simulate module assembly at the level of individual nucleotides. A better approach is to use an abstract model to predict the assembly process with the modules treated as rigid bodies. This section will present earlier such models as a background to my own model described in Chapter 3.

2.2.1 Wang tiles

Introduced by Hao Wang in 1961 [18], s-called *Wang tiles* are

2.2.2 The algorithmic tile assembly model

In his 1998 thesis Erik Winfree showed the usage of the double-crossover (DX) motif to create regular arrays [11]. These tiles behave like Wang tiles[18] and

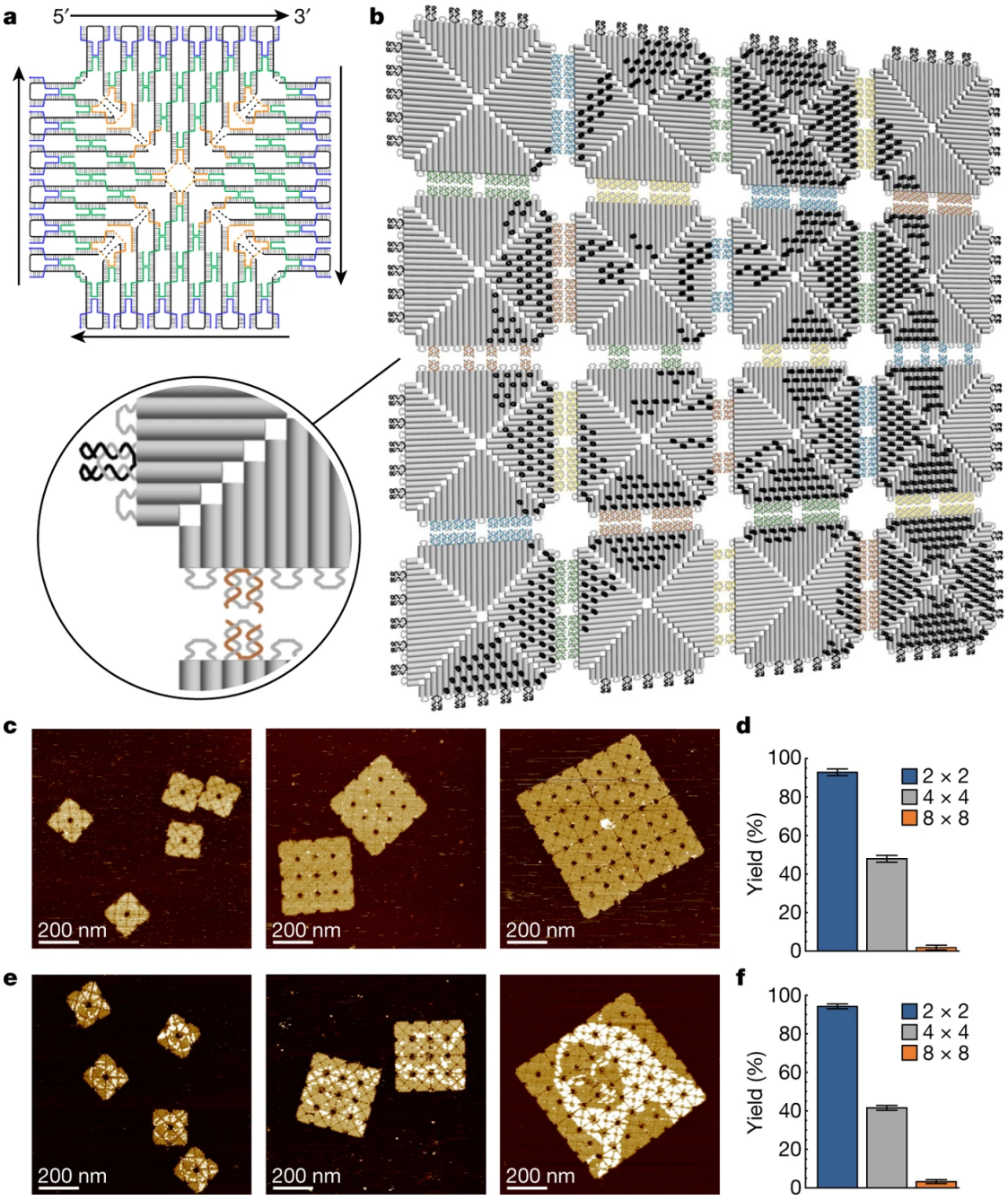

Figure 2.4: Adapted from [13]

do not allow rotations or reflections. Winfree also investigated the possibility of using such tiles for computation [19].

Co-operative binding

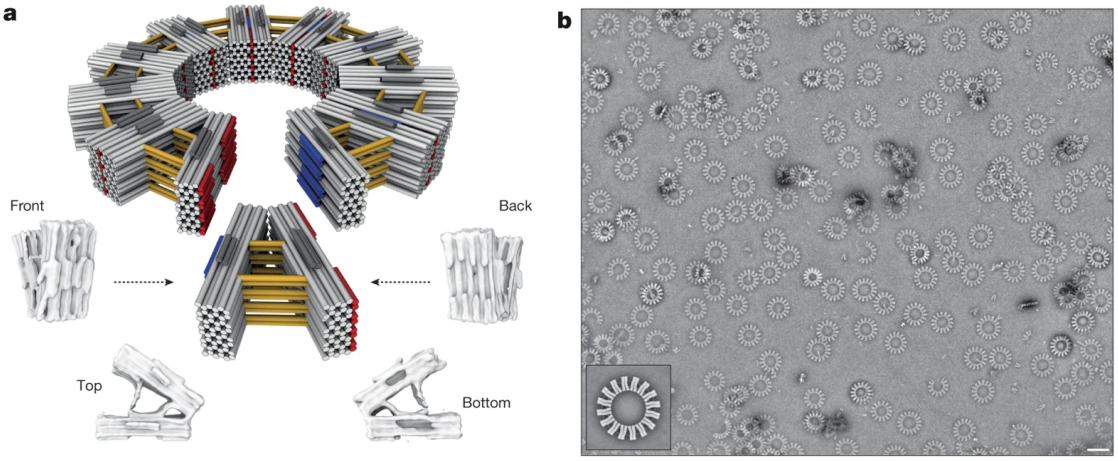

Figure 2.5: Adapted from [14]

2.2.3 The polyomino model

The main inspiration for the polycube model presented in Chapter 3 is the polyomino model [20][21]. The main difference to aTAM is that polyomino tiles are allowed to rotate. They also have a constant binding strength (in other words, it is a temperature-1 model).

Polyominoes are abstract structures composed of one or more squares connected by their edges. The polyomino assembly model developed is similar in assembly to the later experimental micro-meter scale tile designs by Tikhomirov [13] described in Section 2.1.3. Compared to Wang tiles[18], the edge binding does not need to be between edges of the same colour but can be specified by a more complex interaction matrix. Furthermore, the tiles can be rotated to bind, creating further possibilities for symmetries.

I aim to build on Johnston's work on polyominoes to explore the properties of such self-assembling systems in three dimensions; the self-assembly of *polycubes*. With such a model in place, it should be possible to design, through shape complementary in DNA origami [14], kissing interactions in RNA origami [7], or other methods, robotic modules with the interfaces necessary to assemble into the desired polycube shape. My current progress within this project is covered in Chapter 3.

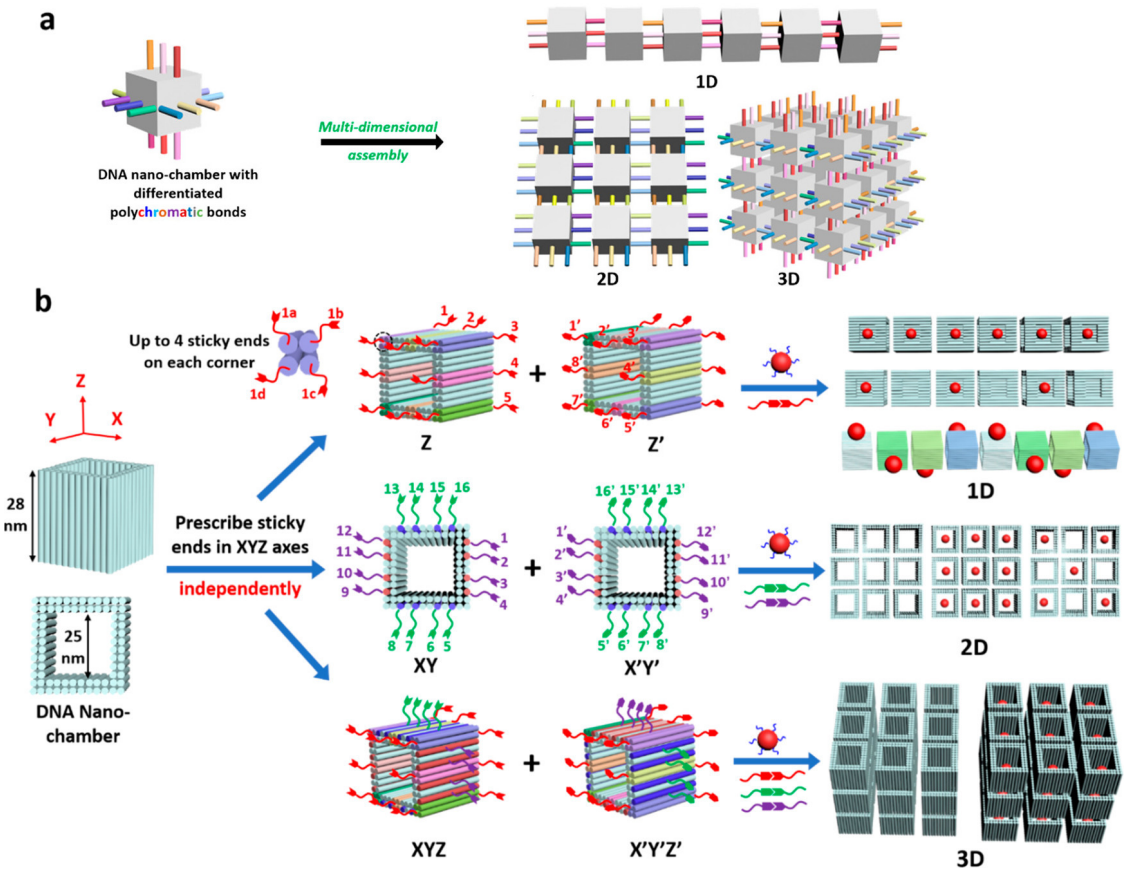

Figure 2.6: DNA nanochambers, adapted from [16]

2.2.4 Patchy particles

2.3 Algorithmic Information Theory and input-output maps

If you have a monkey pressing random keys on a typewriter, you would expect it to produce every string of length N with equal probability (assuming the keystrokes were indeed truly random). With k keys on the keyboard, the probability for any string of length N is then k^{-N}. For example, the title of this thesis, while unlikely to appear randomly, would be equally as probable as any other 50-character string, see the three example strings below:

1 DESIGN AND MODULAR SELF–ASSEMBLY OF NANOSTRUCTURES
2 SHWDRVWKFORWJDOEXOZLSBNREKC Z VSDJJF ROKFYRVMIUI
3 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

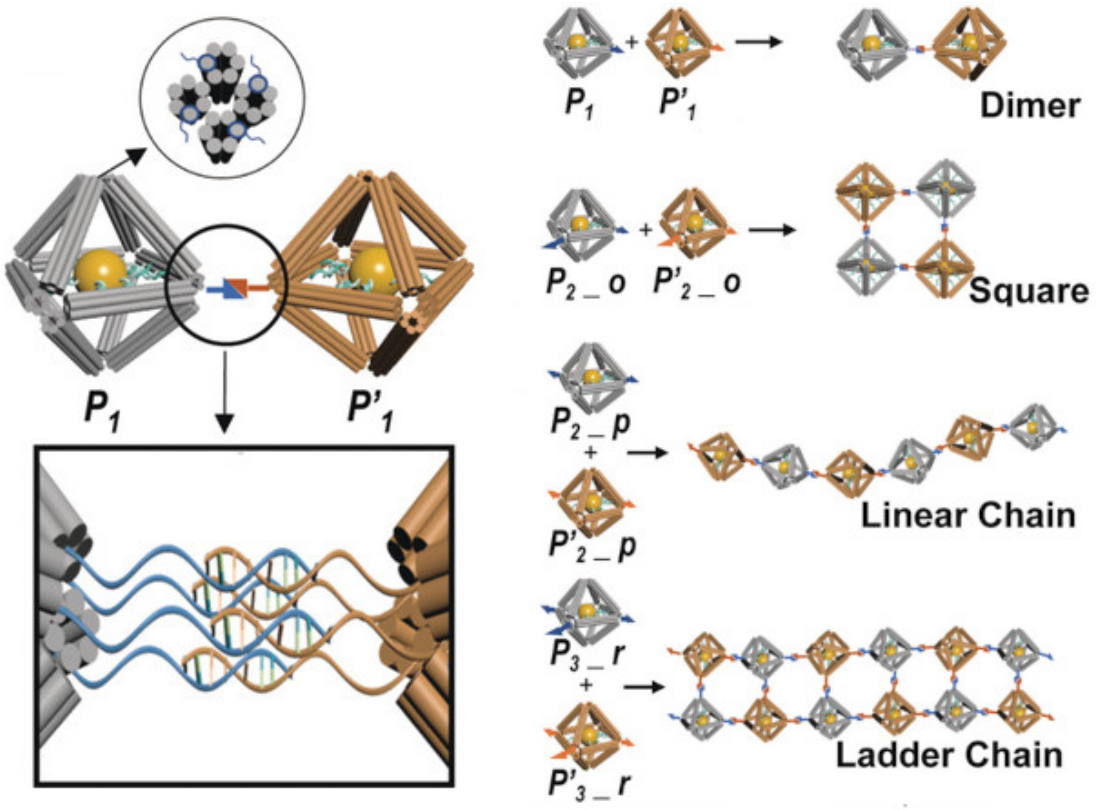

Figure 2.7: Octahedral DNA origami frames, adapted from [17]

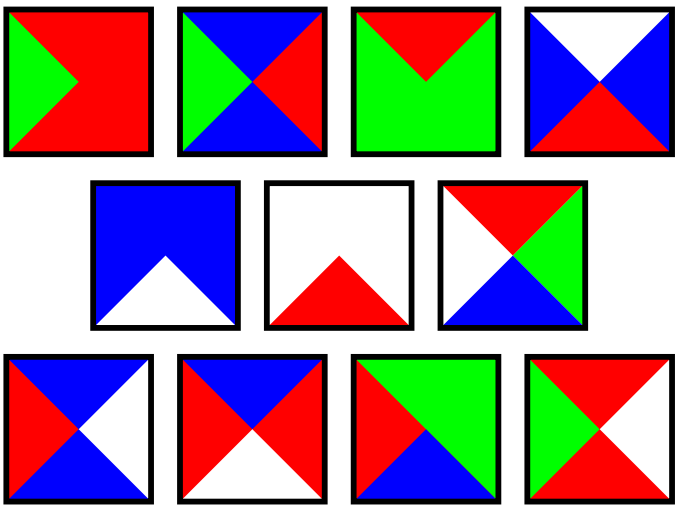

Figure 2.8: Wang tiles

However, some strings can have a description shorter than a full listing of the letters it contains. For example, string number three above could be described simply as "Print `A`, 50 times", or if we use the C programming language:

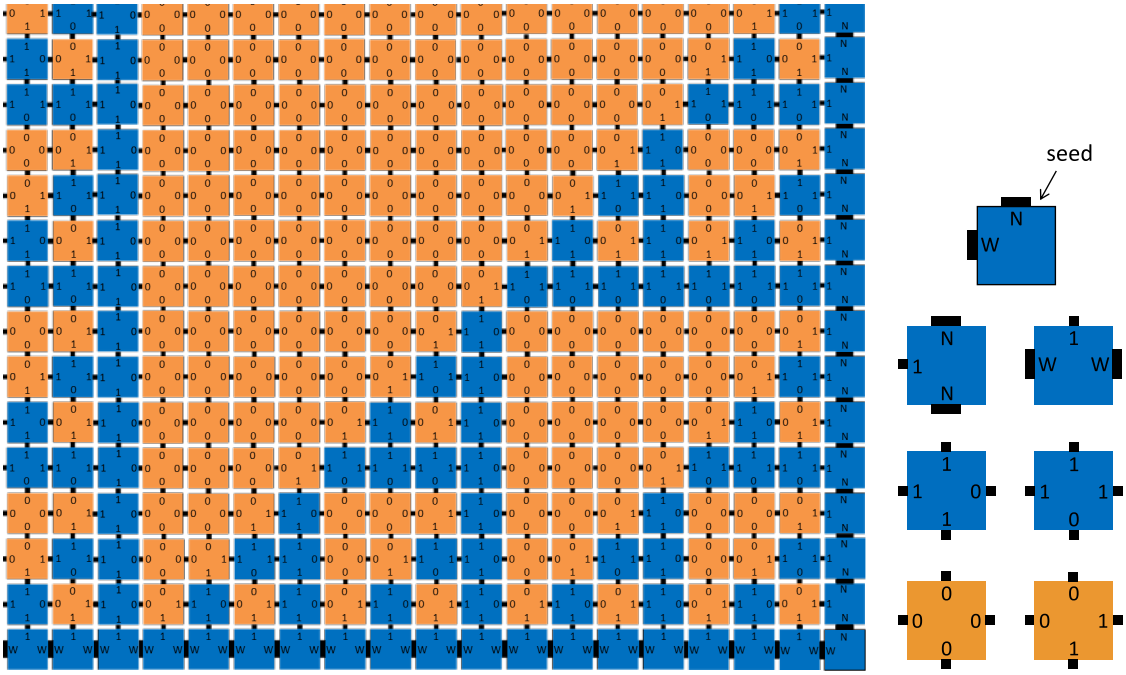

Figure 2.9: Algorithmic self-assembly model

```
for(int i=50; i--;) printf("A");
```

There are also a number of possible valid variations of the code above, with different variable names and coding conventions, all producing the same output. In other words, not only is the description shorter than writing out the full string but multiple inputs map to the same output, increasing the probability of the output further.

The concept of using the shortest possible computer program that can describe an object (for example, a binary string) to determine its complexity is central within the field of Algorithmic Information Theory (AIT) and is called Kolmogorov complexity (or Solomonoff–Kolmogorov–Chaitin to give full credit) [22]. More specifically, the Kolmogorov complexity $K(x)$ of an output x is the length of the shortest program that generates x on a Universal Turing Machine (UTM).

AIT includes the *coding theorem*, introducing lower and upper bounds for the probability $P(x)$ of generating a binary string x as $2^{-K(x)} \leq P(x) \leq 2^{-K(x)+\mathcal{O}(1)}$. In other words, low-complexity outputs are exponentially more likely to generated by random input compared to high-complexity outputs. This could be compared

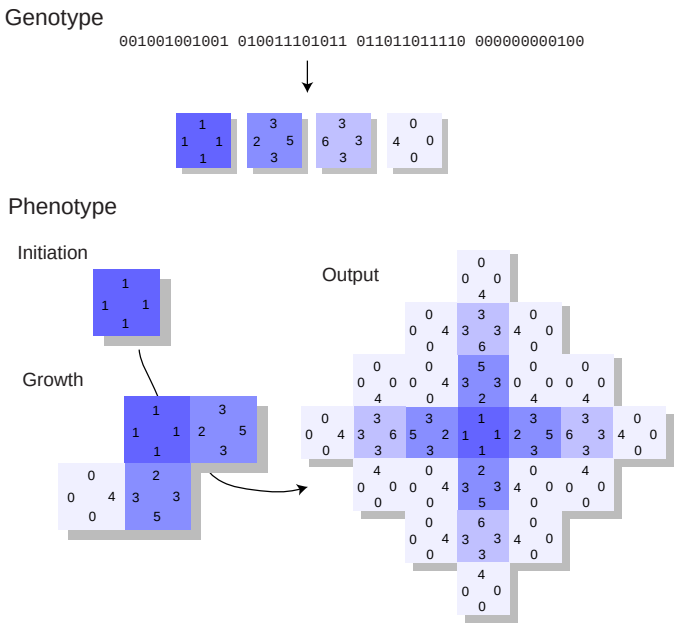

Figure 2.10: Illustration of the polyomino assembly model used by Johnston. A *genotype*, in the form of a ruleset of possible tiles, encodes for a polyomino *phenotype*, grown stochastically from an initial seed tile. Image adapted from [21].

to how, intuitively, there are likely more programs generating the "simple" string number 3 above compared to the randomly generated number 2.

However, finding the shortest program for an output is far from trivial (in general, it is in fact *uncomputable*, due to the *halting problem*). Fortunately, Dingle at al [23] were able to derive an upper bound to the probability using a computable approximation $\widetilde{K}(x)$ of the Komologrov complexity:

$$P(x) \lesssim 2^{-a\widetilde{K}(x)-b}$$

where a and b are constants that depend on the input-output map used (but are independent of x).

3

Modular self-assembly of polycubes

Contents

As introduced in the previous chapter, there is an increasing interest within the field of DNA nanotechnology to create finite-sized multi-component objects. While some coarse-grained tile models exist, there remains a need for methods to quickly explore the assembly of multi-component 3D structures. This chapter describes my polycube model and details how it can be used to sample a large number of assembly rules, showing that some polycube shapes are significantly more common than others. The following chapter will show how to obtain the simplest assembly rule for any given polycube shape.

3.1 The polycube model

A polycube consists of a number of equally-sized cubes connected by their neighbouring faces; a three-dimensional analogue to how polyominos are squares connected by their neighbouring edges. In the model presented here, a polycube is stochastically

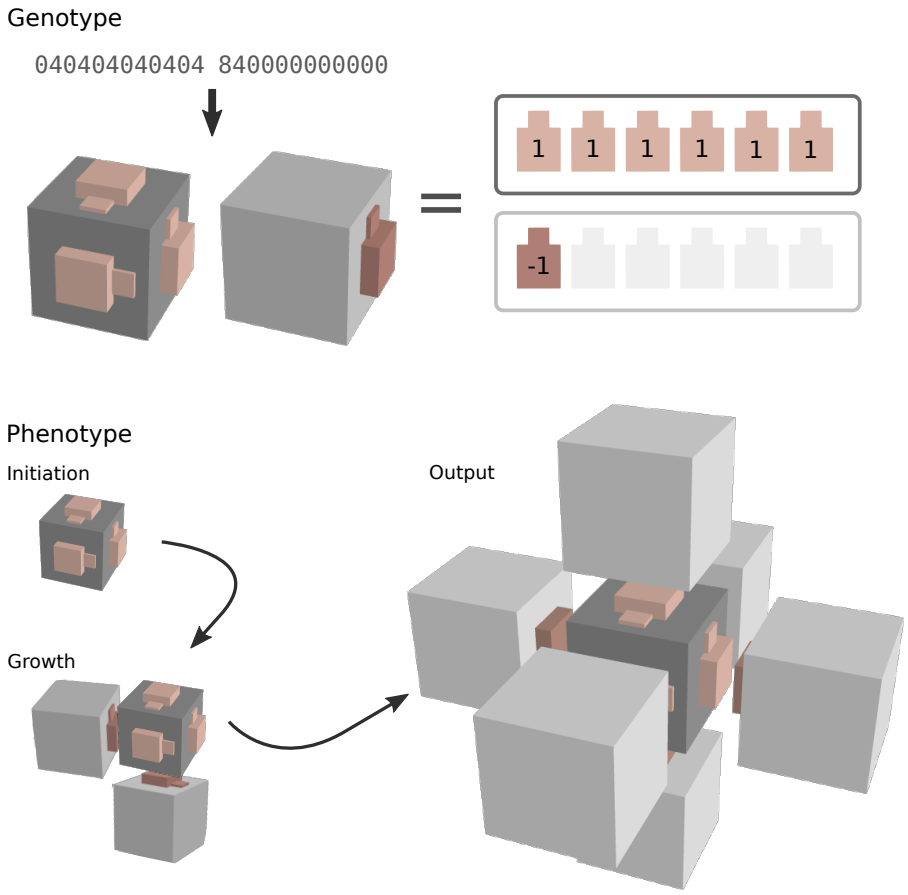

Figure 3.1: Illustration of the polycube assembly model. Compare this to the polyomino model in Figure 2.10. The top of the figure shows the genotype in the form of a rule with two species one colour. The rule is visualised in three different ways. The first is a hexadecimal representation, with each two digits coding for a patch and a total of 12 digits describing a species. The second representation is of the two species shown as 3D cubes. The first species has colour=1 on all patches, while the second has a single patch with colour=-1. Since the polycube is symmetric, the patch rotations do not matter and are all kept at 0.

self-assembled according to a specified rule, where a rule is a set of available species. Each cube species describes a type of cube that can be present in the polycube; cubes belonging to the same species are always identical.

Each species has six patches; one on each face of the cube, and each patch has a "colour" and an orientation. The colour is indicated by a signed integer and the orientation is one of four possible rotations: (0), ($\frac{\pi}{2}$), (π) or ($\frac{3\pi}{2}$). A patch can bind to another patch if and only if they have the opposite colour and the same orientation. If the patch color is zero, the patch is considered empty and will not bind to anything. The model can also be expanded to more complicated

colour interaction matrices or changed to pair odd integers with each subsequent integer as in the polyomino model[20][21] described in Section 2.2.3.

The stochastic self-assembly of a polycube starts by placing a cube from one of the available species as a seed at the origin. If the assembly mode is set to be seeded, it will always use the first species in of rule. If the assembly mode is stochastic, the species of the seed is chosen at random.

Each possible neighbour to the placed cube, given the cube's patches, is then added to a list of possible moves. For the next step, a possible move is chosen at random, and the rule is searched in a random order for a species fitting the move. Cubes can be rotated to fit, and if a fitting species is found, the corresponding cube is added. If there is no fit, the move is discarded. Moves are processed until the list of moves is empty, or the polycube grows beyond a specified size, at which point it is considered unbounded. Figure 3.2.a) shows an example of an unbounded structure tiling the plane using two species

To determine if the rule is deterministic, the assembly is repeated n_{times} times (default 100) and the outputs are compared for equality (allowing rotation). Figure 3.2.b) shows a non-deterministic structure, where the blue "neck" species can bind to itself. Thus depending on how many cubes from the blue species manage to attach before a green species cube binds to stop the growth. Both species are as likely to bind, so the probability of assembling a neck with l blue cubes is 2^{-l}.

It could be argued, instead of first picking a random move and then randomly trying all available species to find a fit, that one should pick both a move and a species at random until a fit is found. While this would take a longer time, it would avoid biasing the assembly toward unlikely assembly results, where a move is picked that would otherwise usually be blocked by more likely surrounding cubes. However, since only deterministic and bounded rules are of interest, this would only affect the end result in the cases where the bias is strong enough and n_{times} is low enough to falsely make the rule seem deterministic.

For an illustration of the model, see Figure 3.1. The example in the figure is a three-dimensional "cross" structure created from a rule of size 2. The initial seeding

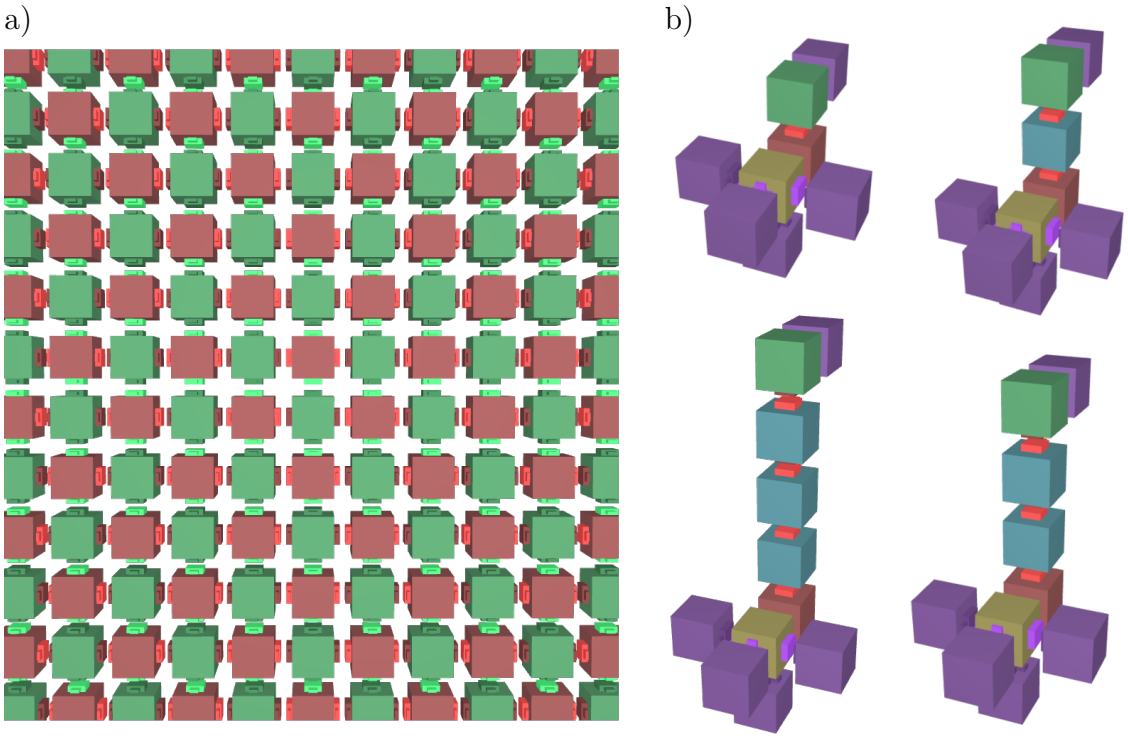

Figure 3.2: Examples of undefined assemblies. a) Unbounded assembly that tiles the plane using two species 05050a08000085858a880000, b) An undeterminisic assembly of a "giraffe duck" with a neck that can have a new length each time it is assembled

cube belongs to the first species, enabling six additional cubes, all belonging to the second species, to bind at each patch. The bounds are made since the patch colours 1 and-1 are opposites. After all six outer cubes have bound, there are no remaining possible moves, and thus the polycube stops growing. Since the growth stops, this particular polycube is bounded at a size of seven cubes. Furthermore, since the rule gives the same polycube every time it is evaluated, the polycube is deterministic.

The polycube assembly model has been implemented in two versions: one browser implementation for outreach activities and accessible visualisation and one C++ implementation for fast rule evaluation. Using the C++ implementation, large sets of random rules have been evaluated, and the resulting polycubes examined and categorised. This has also been repeated for different values of rule size and colour limits.

See Figure 3.4 for a heat map of the frequency of rules of a certain size creating polycubes of a certain size. Notice how only three different polycube sizes were found

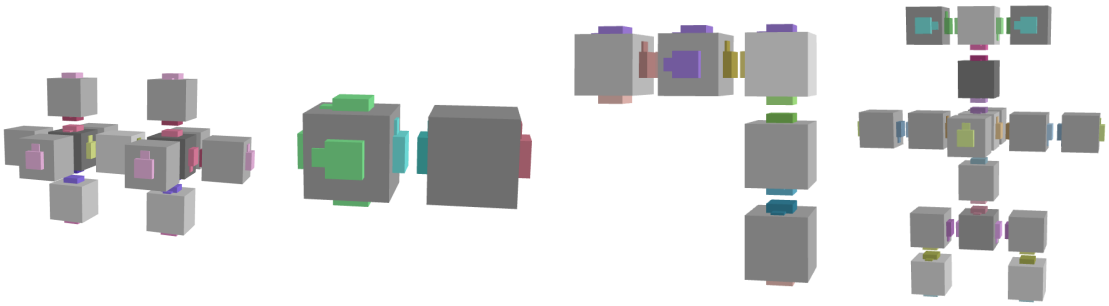

Figure 3.3: Polycube versions of the conceptual DNA Robotics designs. Compare to Figure **??**. From left to right, the size of the rule required to specify each polyomino is: 4, 2, 5 and 11. Note how the third polyomino (L-shaped) requires a larger rule than the first (double cross), although it is smaller in size. This is because each cube in the L-shaped nanobot needs to be unique, while the double-cross-shaped first nanobot consists of two identical parts. If we only wished to reproduce the polycube shape, the rulesets could be minimised further.

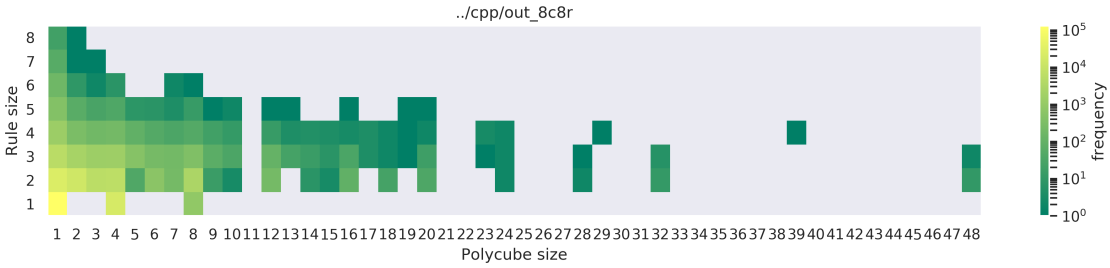

Figure 3.4: Heat map of rule size vs polycube size, for a simulation with a maximum of 8 species and a limit of 8 colours, evaluating 2.5 million random rules. Note that the frequency scale is logarithmic.

for one species. This is understandable when inspecting the first column in Figure 3.5; using only one species, you cannot create a bounded structure of any other size.

For larger rule sizes, polycubes can have any form, as shown in Figure 3.3, but upon inspection of the larger polycube sizes (and small rule sizes) from Figure 3.4, the polycubes all seem to be highly symmetrical, as shown in 3.5.

Figure 3.5: Example of polycubes grown from 1, 2 and 3 species. Although any polycube can be encoded into a rule, the larger polycubes that have small rules tend to be symmetrical. This agrees with Johnston's polyomino model in that they have low complexity

3.2 Sampling the space of assembly rules

$$I_{n_c,n_t} = (4(1+2n_c))^{6n_t}$$

Trying all inputs in a brute-force approach is impossible. Even if we limit both the number of allowed colours n_c and the amount of species n_t to eight, the space of possible inputs, with four patch orientations and six patches per cube, is $I_{8,8} \approx 9 * 10^{87}$. But, while this is too large to sample fully, we can still get an idea of how likely it is for an input to map to a certain output by taking samples of the space of possible input rules.

This was done by uniformly sampling and assembling one billion random rules from $I_{8,8}$. Rules growing larger than 100 cubes were discarded as unbounded, while those remaining bounded were re-assembled 15 times to ensure they assembled deterministically. Deterministic and bounded output was then grouped by their shapes, counting the number of times each given phenotype occurs. For each rule found to produce a given phenotype, the number of colours is multiplied by the number of species in the rule, producing a measure of the rule size. The smallest such rule size is then used as a proxy for the phenotype complexity.

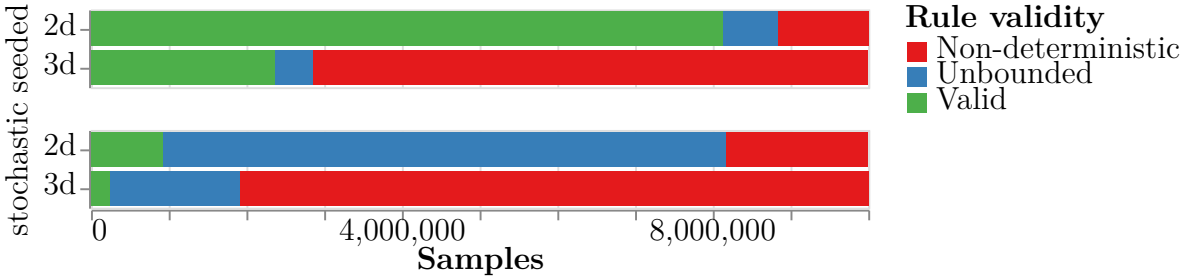

Figure 3.6: Proportion of valid rules in different sampling datasets

3.3 Complexity bias

As can be seen in Figure 3.8, there is a clear log-linear relationship between the probability of finding a rule that assembles into a particular structure and the information needed to specify the structure, as predicted in [23, 24].

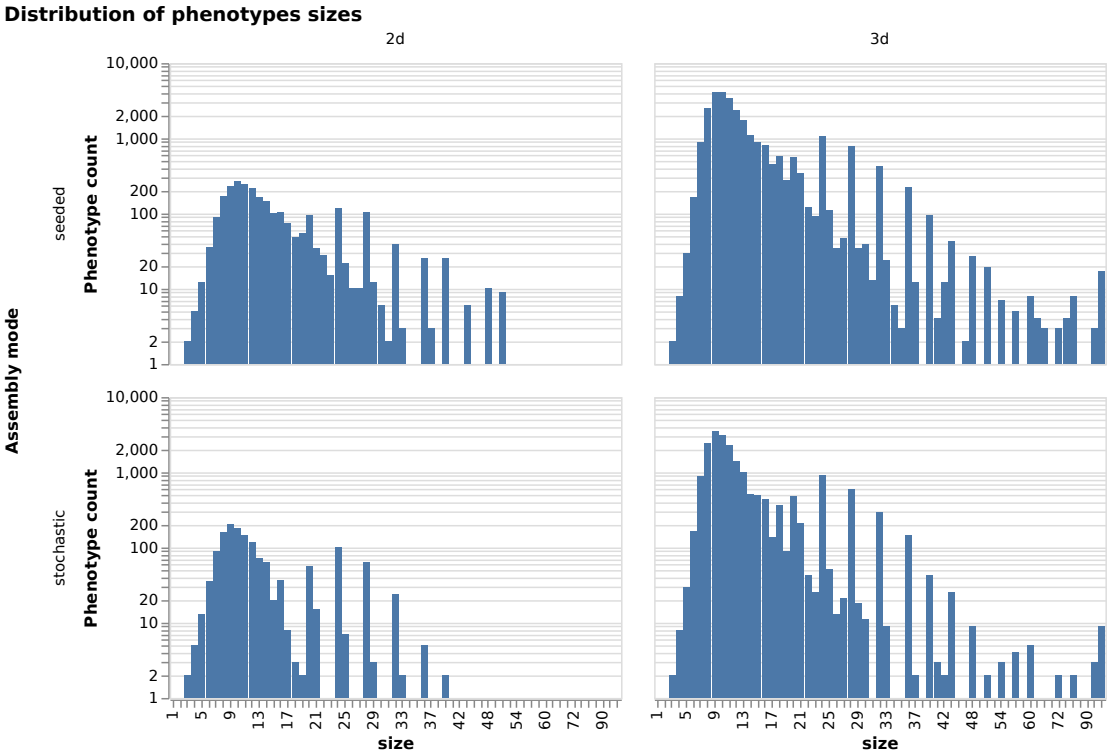

Figure 3.7: Distribution of phenotype sizes.

Stochastic sampling tends to have a constant high number of cube types. Since any of the cube types can be used as a seed, they all need to grow into the same structure every time. Thus they all need to be included.

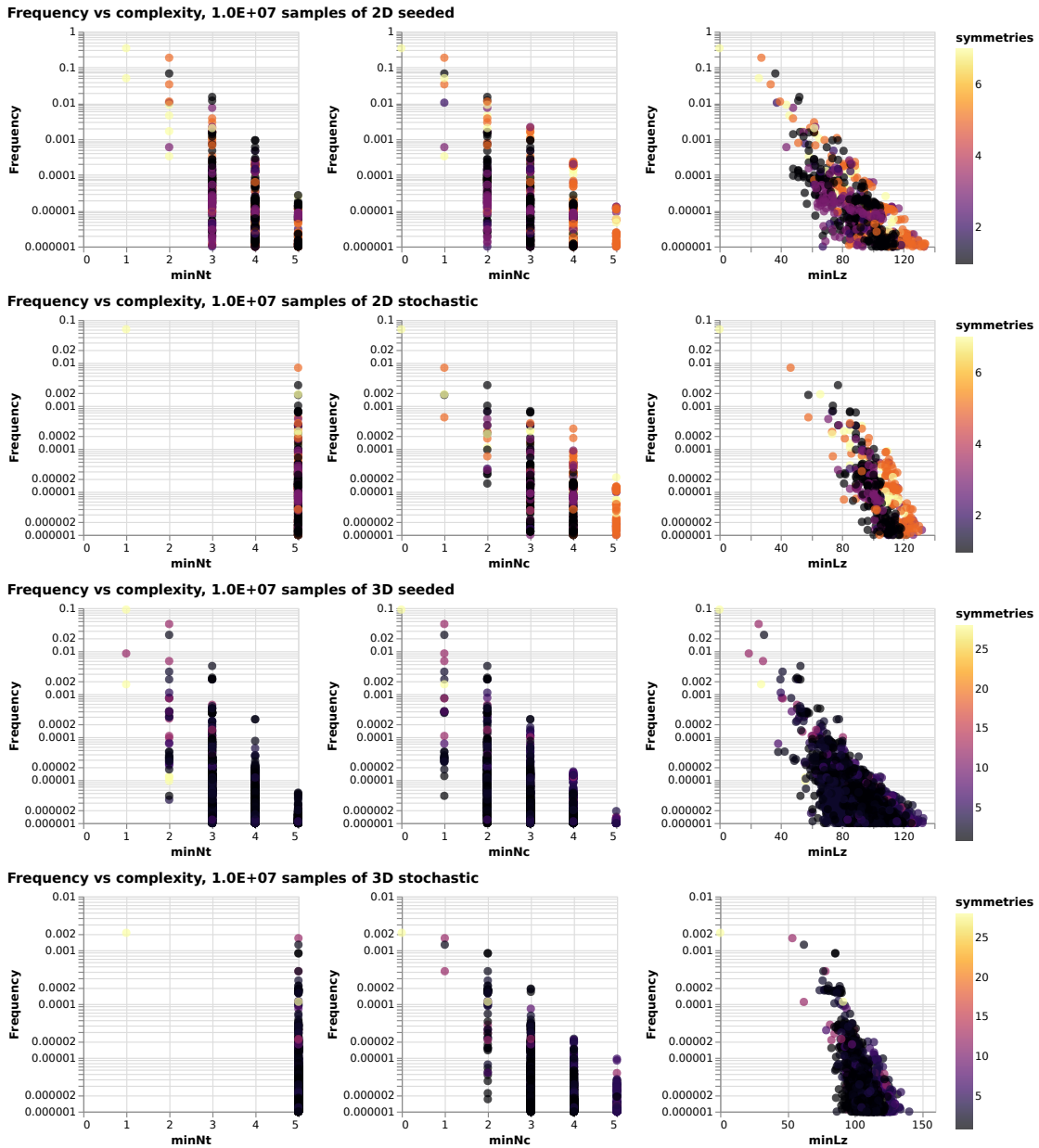

Figure 3.8: Frequency vs complexity

4

Designing polycube assembly rules

Contents

Fully addressable is the easy solution

Could also sample the whole input space as in previous chapter

But is there a better way?

4.1 Satisfiability solving

Building upon a method for determining patchy particle interactions for unbounded structures [25] it is possible to also formulate and solve satisfiability problems for the bounded polycube structures.

Clauses (i)-(vii) are the same as in [25], while the remaining clauses have been added, together with variables x^D, x^A and x^O, to include torsional restrictions.

Interaction matrix is fixed (compared to [25]).

Figure of topology graph. Figure of particle schematic

Add giraffe-duck-like example of when SAT solver gives UND solution

Variable	Description
$x^A_{l,p,o}$	Patch p at position l has orientation o
$x^B_{c_i,c_j}$	Colour c_i is compatible with colour c_j
$x^C_{s,p,c}$	Patch p on cube type s has colour c
$x^D_{p_1,o_1,p_2,o_2}$	Patch p_1, orientation o_1 binds to patch p_2, orientation o_2
$x^F_{l,p,c}$	Patch p at position l has colour c
$x^O_{s,p,o}$	Patch p on cube type s has orientation o
$x^P_{l,s,r}$	Position l is occupied by cube type s rotated by r

Table 4.1: SAT variables and descriptions

Id	Clause	Boolean expression
(i)	$C^B_{c_i,c_j,c_k}$	$\neg x^B_{c_i,c_j} \vee \neg x^B_{c_i,c_k}$
(ii)	C^C_{s,p,c_k,c_l}	$\neg x^C_{s,p,c_k} \vee \neg x^C_{s,p,c_l}$
(iii)	C^P_{l,s_i,r_i,s_j,r_j}	$\neg x^P_{l,s_i,r_i} \vee \neg x^P_{l,s_j,r_j}$
(iv)	$C^{BF}_{l_i,p_i,c_i,l_j,p_j,c_j}$	$\left(x^F_{l_i,p_i,c_i} \wedge x^F_{l_j,p_j,c_j}\right) \Rightarrow x^B_{c_i,c_j}$
(v)	$C^{rotC}_{l,s,r,p,c}$	$x^P_{l,s,r} \Rightarrow \left(x^F_{l,p,c} \Leftrightarrow x^C_{s,\phi_r(p),c}\right)$
(vi)	C^{alls}_{s}	$\bigvee_{\forall l,r} x^P_{l,s,r}$
(vii)	C^{allc}_{c}	$\bigvee_{\forall s,p} x^C_{s,p,c}$
(iix)	C^{O}_{s,p,o_k,o_l}	$\neg x^O_{s,p,o_k} \vee \neg x^O_{s,p,o_l}$
(ix)	$C^{DA}_{l_i,p_i,c_i,l_j,p_j,c_j}$	$\left(x^A_{l_i,p_i,c_i} \wedge x^A_{l_j,p_j,c_j}\right) \Rightarrow x^D_{p_i,c_i,p_j,c_j}$
(x)	$C^{rotO}_{l,s,r,p,o}$	$x^P_{l,s,r} \Rightarrow \left(x^A_{l,p,o} \Leftrightarrow x^O_{s,\phi_r(p),o}\right)$

Table 4.2: SAT clauses. (i) Each colour is compatible with *exactly one* colour. (ii) Each patch has *exactly one* colour. (iii) Each lattice position contains a single cube type with an assigned rotation. (iv) Adjacent patches in the lattice must have compatible colours. (v) Patches at a lattice position are coloured according to the (rotated) occupying cube type. (vi) All N_t cube types are required in the solution. (vii) All N_c patch colours are required in the solution. (iix) Each patch is assigned *exactly one* orientation. (ix) Adjacent patches in the target lattice must have the same orientation. (v) Patches at a lattice position are oriented according to the (rotated) occupying cube type.

4.2 Finding the minimal assembly rule

The method presented here uses a SAT solver to determines if a provided polycube shape is satisfiable for a given number of cube types N_t and colours N_c. By iteratively ruling out lower values of N_t and N_c, a minimal solution can be found, as detailed in Figure 4.1. It is also possible to generate and compare alternative solutions of varying complexity.

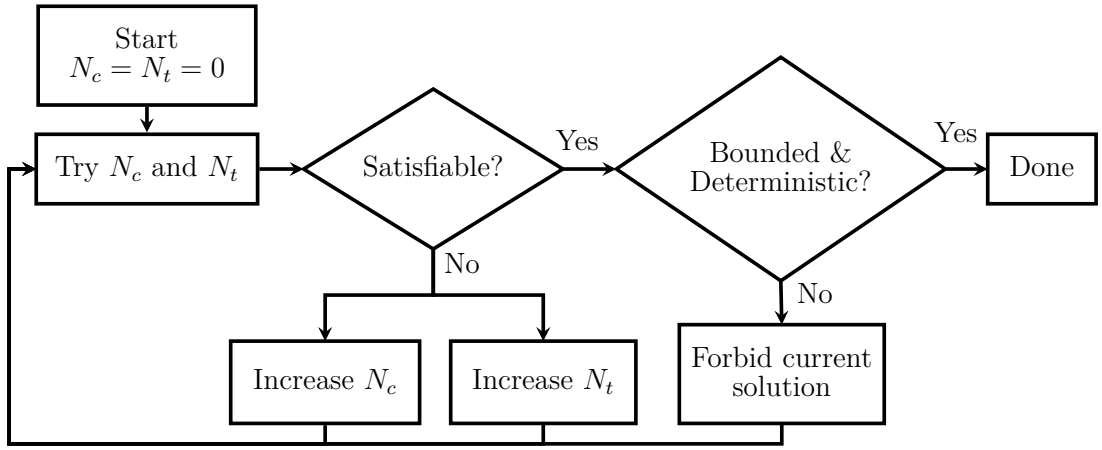

Figure 4.1: Algorithm for finding the minimal solution using SAT. Even if a solution is found to be satisfiable it might not assemble correctly every time. Additional solutions for a given N_c and N_t are found by explicitly forbidding the current solution. Alternatively, it is possible to use a solver like relsat to obtain multiple solutions.

4.3 Example solves

A process cannot be understood by stopping it. Understanding must move with the flow of the process, must join it and flow with it. (First Law of Mentat)

— Frank Herbert, Dune

5

An introduction to tools for the design and simulation of nucleic acid structures

Contents

While previous chapters have covered modular self-assembly on a very abstract level, approximating the modules as simple cubes or patchy particles, this chapter will introduce tools and methods for designing and simulating individual structures or modules folded using DNA (or RNA).

The following sections will cover a selection of useful design and simulation tools that have been developed over the years, providing context for the presentation of my contributions to the *oxView* tool in Chapter 6.

5.1 Design tools

Designing a DNA origami structure by hand would be a very labourious task for anything but the most simple design. As such, a host of computer-aided design tools have been introduced over the years to make things easier. This section will cover some of the more popular examples.

5.1.1 Lattice-based design tools

The caDNAno design tool [26] and the web-based scadnano[27] it inspired, allows the user to design DNA origami on a lattice of parallel helices.

caDNAno

CaDNAno [26] was introduced in 2009 as a way to simplify 2D and 3D DNA origami designs. It has a graphical user interface, seen in Figure 5.1, where the designer can place virtual helices on an either hexagonal or square lattice in a slice panel (the leftmost panel in the figure). The helices can then be filled in with strands and connected using crossovers in a path panel, seen in the middle of the figure.

Finally, caDNAno is also available as a plugin to the Autodesk Maya software, which enables a 3D visualisation of the design as seen in the render panel to the right in Figure 5.1. However, caDNAno doesn't support Maya versions after 2015 [28].

Scadnano

Scadnano [27] (scriptable caDNAno) is a relatively new design tool, independent from but inspired by caDNAno version 2. The main difference is that Scadnano is completely web-based (not requiring any installation). The python code base is also designed to make it easier to write scripts generating DNA designs.

5.1.2 Top-down shape converters

While tools like caDNAno simplifies bottoms-up design, where the user builds structures from individual strands and nucleotides, a top-down tool can take a target polyhedral shape as input and provide a suitable origami design as output.

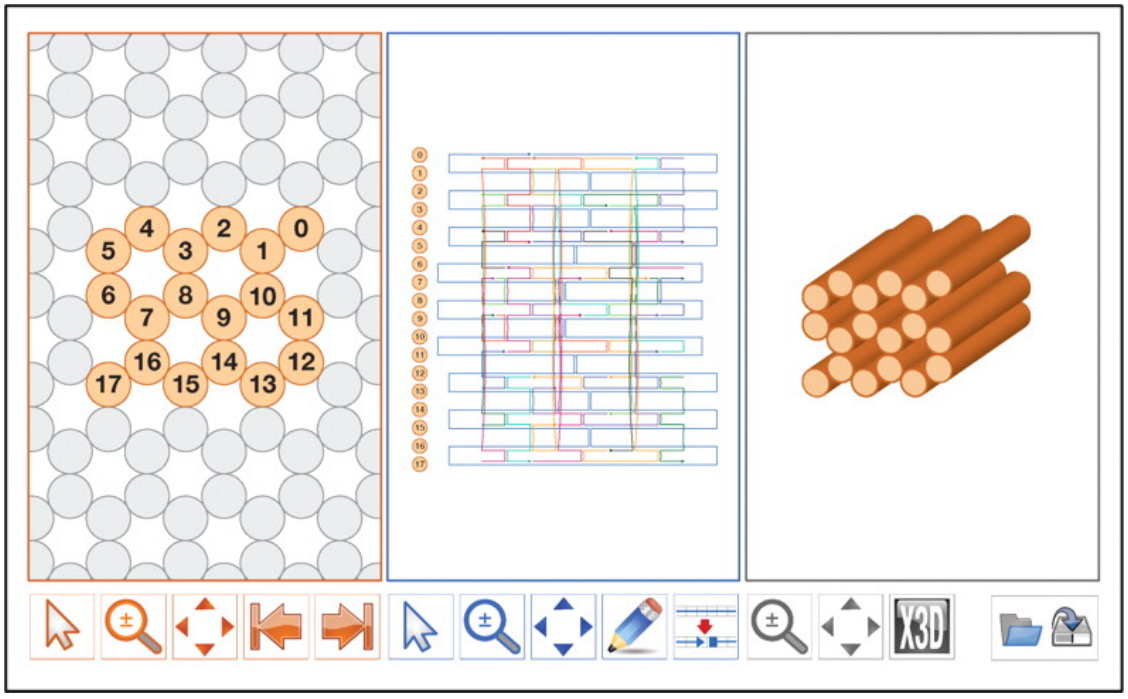

Figure 5.1: The caDNAno design interface, adapted from figure 1.(a) in [26]

BSCOR

DAEDAULS/PERDIX

ATHENA?

Triangulated truss structures

5.1.3 Free-form or hybrid tools

Tiamat

Tiamat is an early free-form design tool introduced in 2009

vHelix

Adenita

A final option for structure editing is Adenita [29]. So far, a version has only been released for beta testers and unfortunately, they do not yet support Linux. Still, after talking to Haichao Miao at the Nantech2019 conference, Adenita sounds like it could become a valuable off-lattice tool for designing and editing DNA nanostructures, with import options from caDNAno and export to oxDNA.

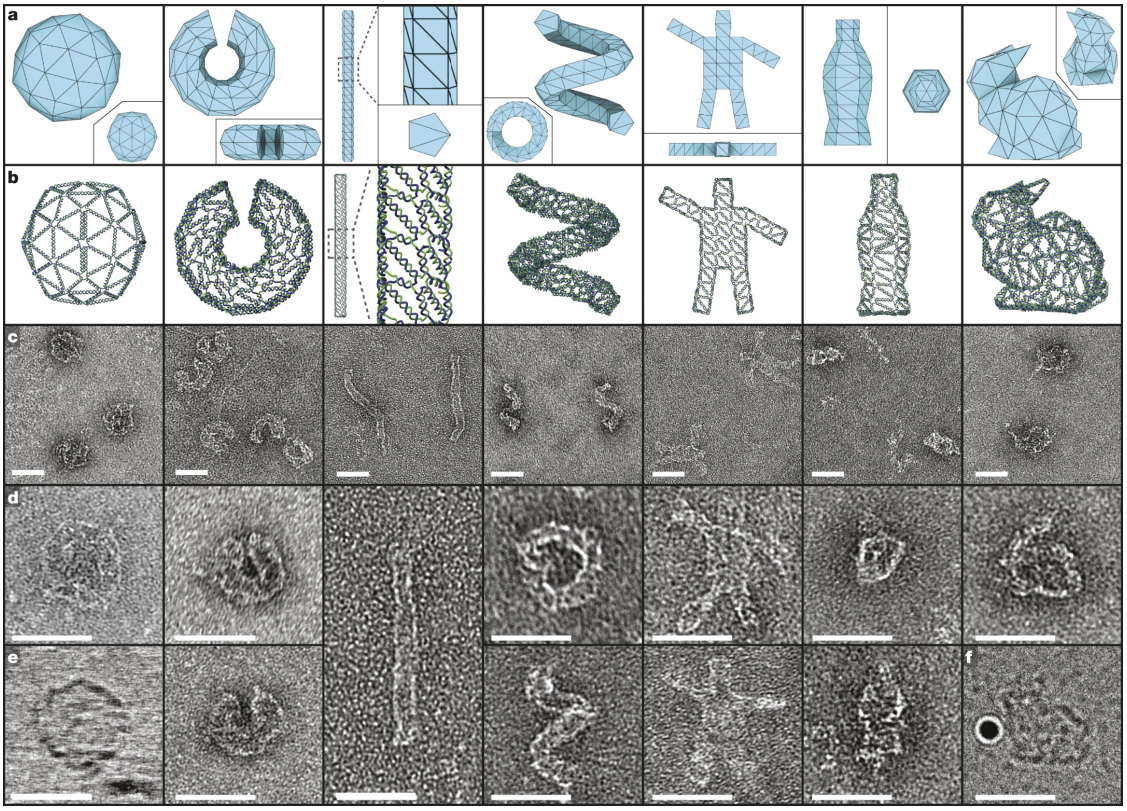

Figure 5.2: BSCOR

MagicDNA

Another method of solving topological issues through rigid-body manipulation was used by Chao-Min Huang [30]. The tool used is not public, so I have not been able to examine it, but the supplementary material of the article shows them using a Matlab script to interactively move and rotate the helix bundles (treated as rigid bodies) into a desired configuration with less stretched bonds.

oxView

The oxView application was developed as part of this thesis project and will be described more in Chapter 6.

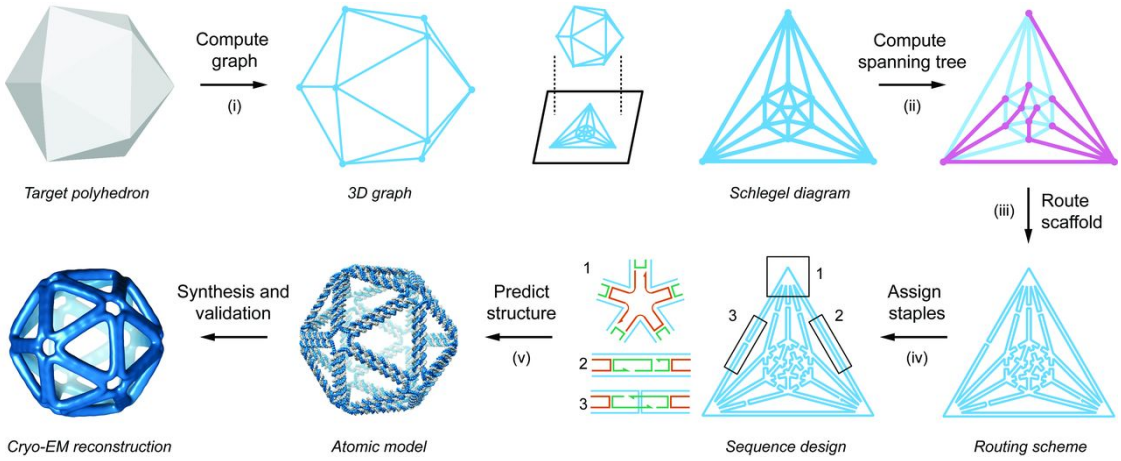

Figure 5.3: daedalus

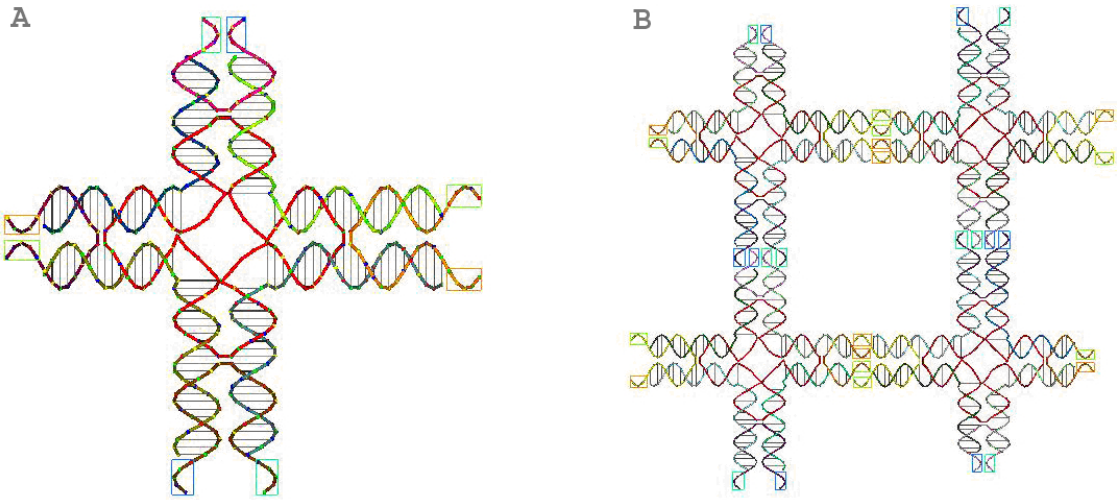

Figure 5.4: Tiamat

5.2 Simulation tools

Simulating a structure can provide insight to understand experimental results, but can also guide decisions at the design stage. Different models

5.2.1 All-atom simulation

Simulation tools such as NAMD[31], use force fields such as AMBER[32] and CHARMM [33] that model interactions between individual atoms. While it is possible to perform atomistic simulations of large DNA origami structures[34],

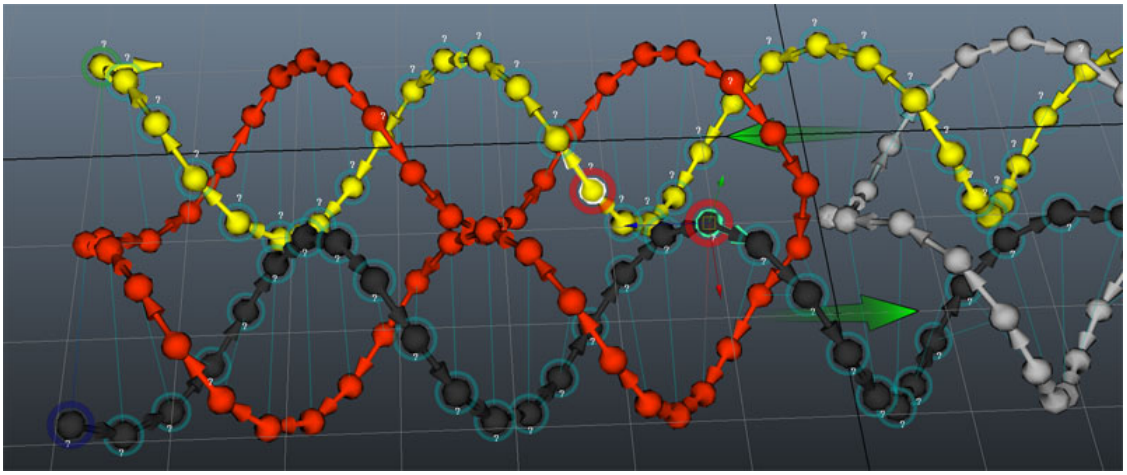

Figure 5.5: vHelix

the simulations take a long time to run and it is unknown how well the models represent DNA thermodynamics [35].

5.2.2 oxDNA/RNA

In 2010, a the coarse-grained simulation software called oxDNA was introduced by Thomas Ouldridge [4]. It simulates DNA on the level of nucleotides and has been shown to model complex origami devices with a generally good agreement with experimental data [36]. In 2014, the DNA model was extended to include RNA by Petr Šulc [37], showing its ability to model a set of common RNA motifs.

Molecular Dynamics (MD) and Monte Carlo (MC) simulation techiniques.

OxDNA was joined by cogli1 for trajectory visualisation.

While oxDNA can be very useful for modelling a structure, it has traditionally not been very accessible for experimentalists.

5.2.3 mrDNA

Another promising feature of mrdna is that it has a spline-based helix representation and is implemented in python. This enables users to easily script edits to the stucture, translating and rotating parts of it before starting the simulation. Thus, topological issues or over-stretched bonds can, with some skill, be resolved even before starting

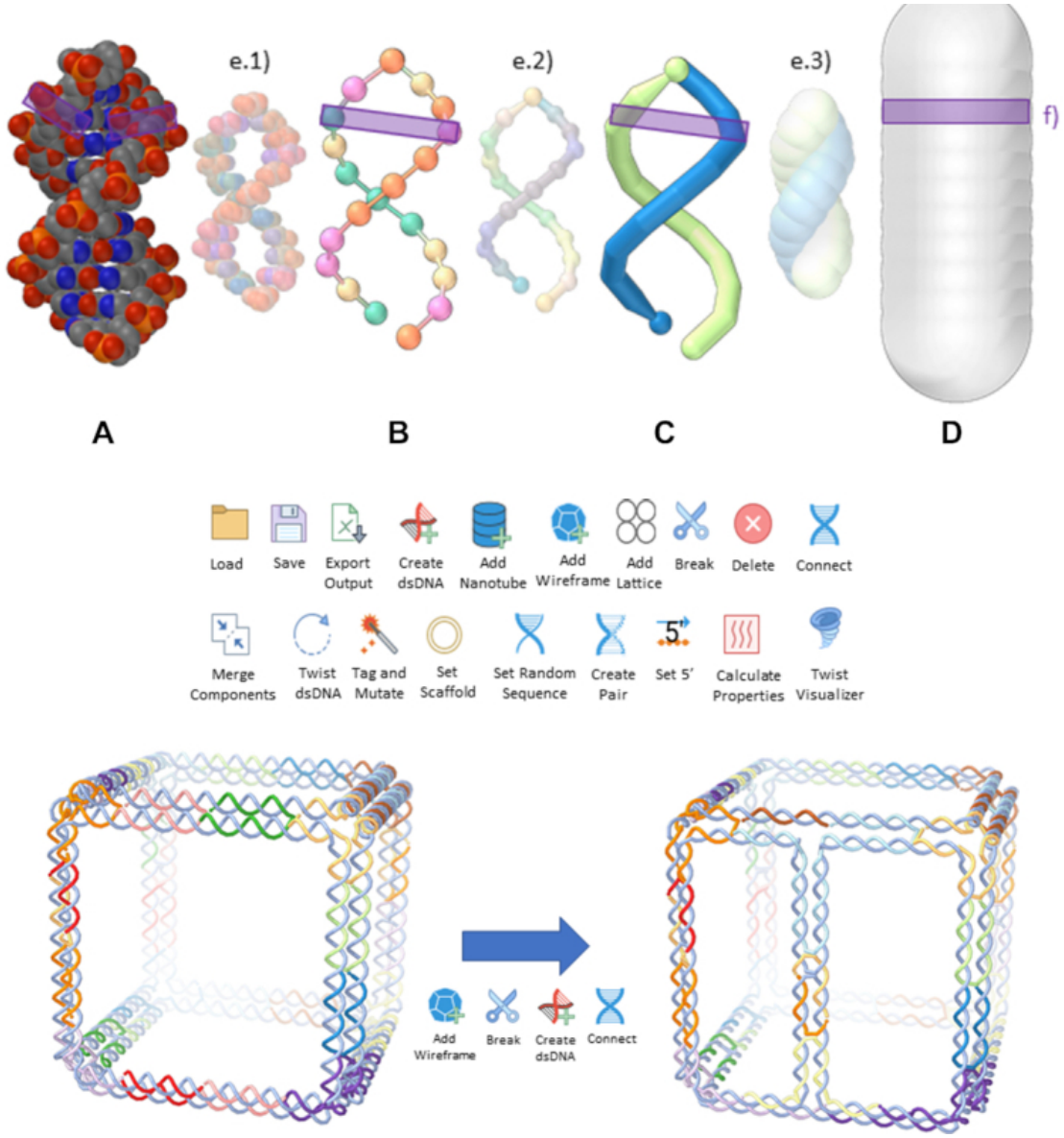

Figure 5.6: Adenita

to simulate. I did, based on this, also create a rudimentary interactive editor interface to mrdna, but it would need a lot of refinement to be externally usable.

5.2.4 Cando

Cando is a finite element modelling framework[42] available through a web server at `https://cando-dna-origami.org`. DNA double helices are modelled as elastic rods (connected by rigid crossovers) that stretch, twist and bend in line with

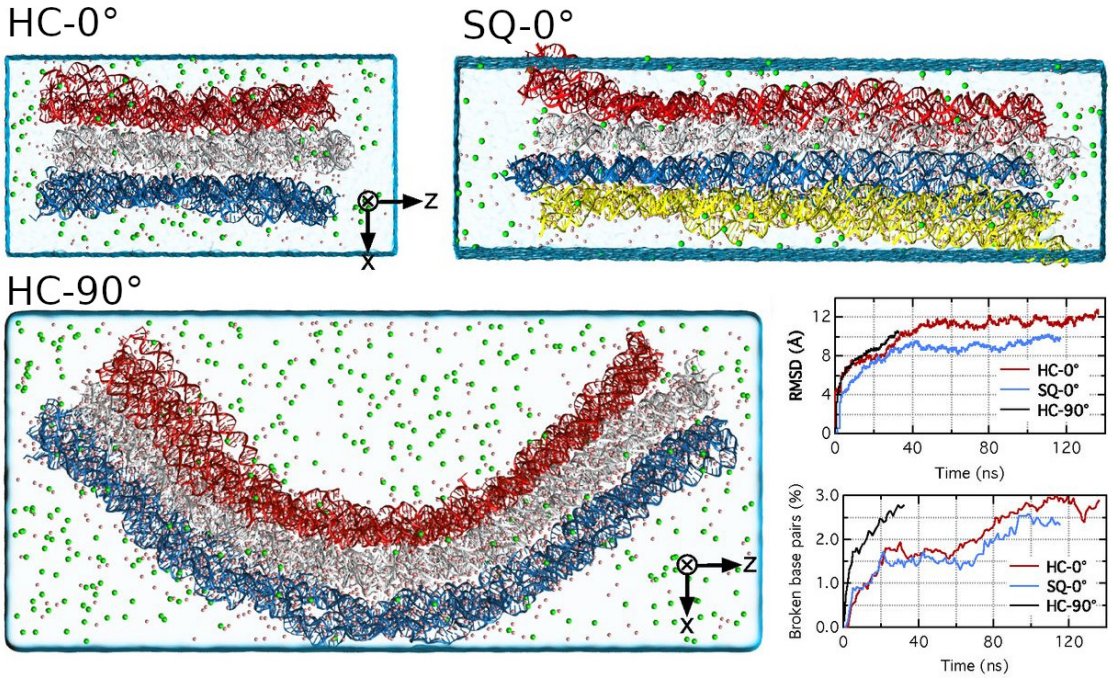

Figure 5.7: All-atom simulation

experimental measurements.

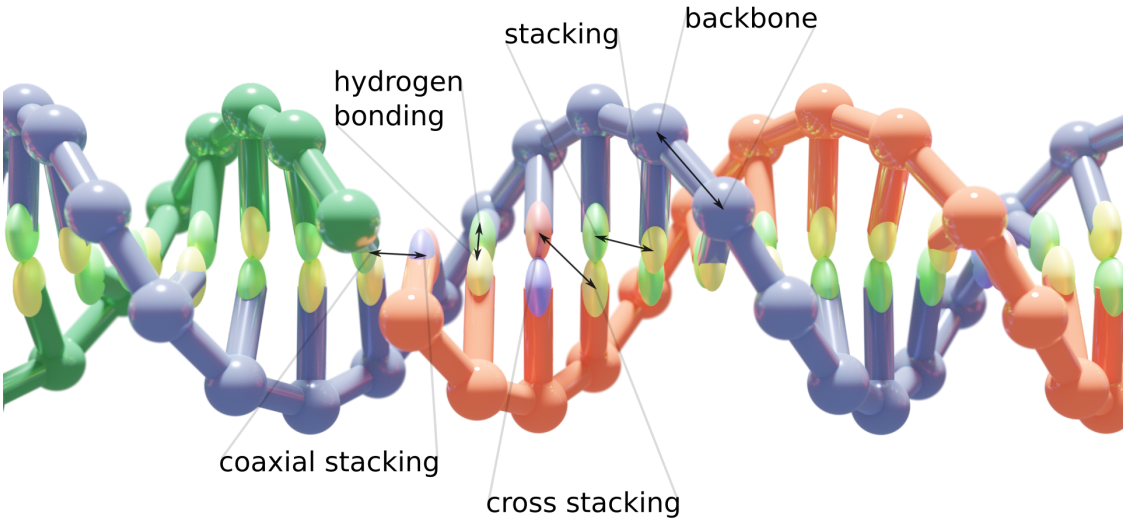

Figure 5.8: The oxDNA model

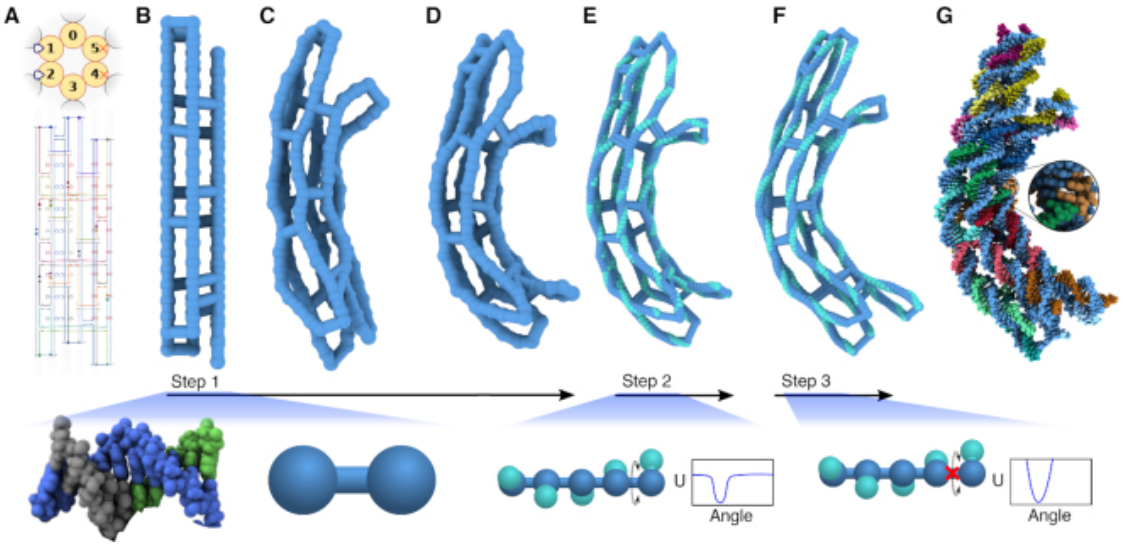

Figure 5.9: MrDNA

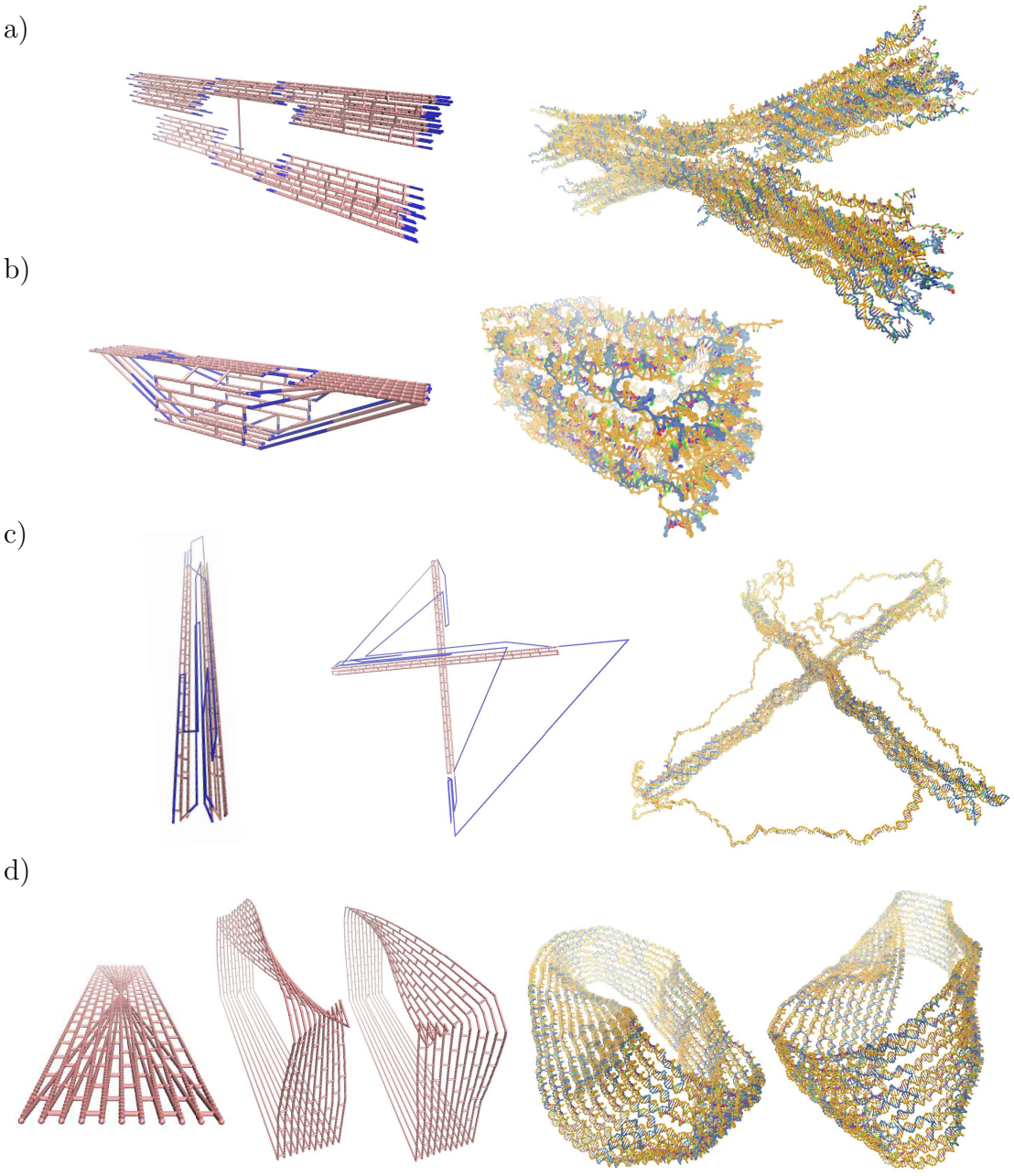

Figure 5.10: Relaxation results for various DNA designs. Each row depicts a new design, with the left-hand side showing the structure as it was drawn in caDNAno (and parsed by mrdna), while the right-hand side is the relaxed structure in oxDNA. Intermediate images are edits done in mrdna. While the switch design[38] in **a)** and the small DNA origami box[39] in **b)** relaxed without any required editing, the tensegrity kite structure [40] in **c)** and the Möbius strip[41] in **d)** benefited greatly from moving selected helices to a position off the lattice before starting the simulation.

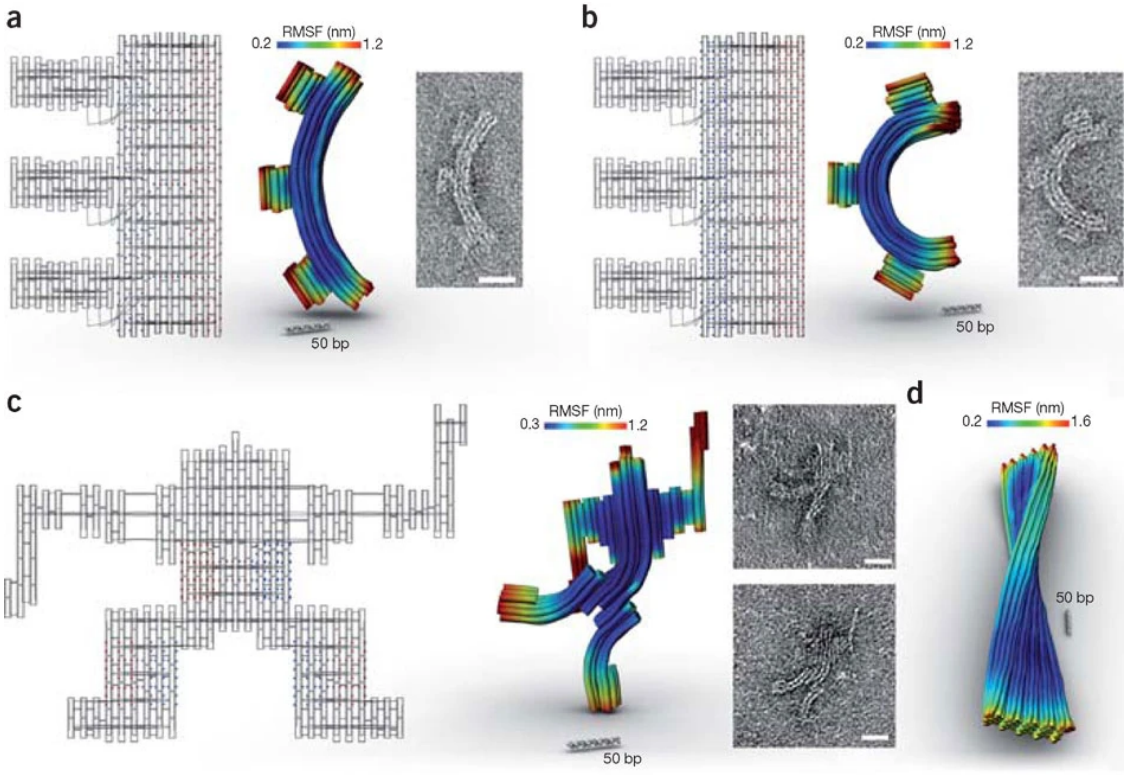

Figure 5.11: Cando

6

Structure design and analysis in oxView

Contents

This chapter contains my results on how to design, simulate and analyze DNA and RNA nanostructures. It starts with an introductory description of earlier approaches to this problem and will then describe how my contributions to the oxView tool have improved the previous methods and enabled new results.

As I started this DPhil, I was tasked with the issue of converting origami designs created in caDNAno so that they can be correctly simulated in oxDNA.

I soon started collaborating with Hannah Fowler from the Doye group at Theoretical Chemistry. She was simulating an extensive collection of old DNA designs, so it was a good opportunity to convert and relax problematic structures, investigating why some structures are more problematic than others.

At this time, there was a python script included in the UTILS directory of the oxDNA repository, but it was not easy to use, and it failed for many structures. At the end of 2018, the tacoxDNA webserver [43] was launched, updating the conversion script and making it more accessible. Still, since caDNAno structures are drawn on a lattice, the resulting oxDNA configurations often had unnaturally extended backbone bonds, requiring time-consuming relaxation.

During a secondment within the Šulc group at Arizona State University in 2019, I contributed to the development of a web-based oxDNA viewer called oxView[44]. Among the main features I added to oxView was a cluster-level rigid-body dynamics option (detailed in Section 6.2) that in many cases speed up the relaxation with orders of magnitude compared to oxDNA relaxation alone. Since then I have been collaborating with the Šulc group to add more features and to make the tool more accessible as a visualizer and editor. For our second oxView publication I rewrote the main parts of the taxoxDNA codebase into typescript, resulting in the *taxoxdna.js* library (`https://github.com/Akodiat/tacoxdna.js`) which oxView uses to import various common design formats automatically.

This section will present my own contributions to oxView, unless otherwise stated, but also I want to acknowledge the work done by Erik Poppleton and Michael Matthies; without them this tool would not exist. Michael was the original oxView developer and has done a great work in enabling live oxDNA relaxations through the *ox-serve* webserver. Erik made oxView able to smoothly render and analyze systems with over a million nucleotides and has created a large set of handy analysis scripts.

6.1 Importing designs

There are a lot of different formats available for DNA origami design; some of the main design tools are covered in Section 5.1. This section will describe how

to use oxView to import different designs.

6.1.1 Basic import

Thanks to the *tacoxdna.js* library I created, the conversion is now straightforward for many designs. The formats listed below can all be imported by simply clicking the *Import* button in oxview. Some additional formats still require the use of the external Tacoxdna webserver[45] however.

Importing caDNAno files

Select the caDNAno JSON file to import, making sure that *caDNAno* is selected as file format. Next, select the correct lattice type; either *Square* or *Hexagonal*. Optionally, input a sequence to assign to the origami scaffold (which will otherwise be random).

Importing rpoly files

Rpoly files are the output from the BSCOR[46] tool (described in Section 5.1.2) for converting polyhedral meshes into DNA origami. Select the rpoly file to import, making sure that *rpoly* is selected as file format. Optionally, input a sequence to assign to the origami scaffold (which will otherwise be random).

Importing Tiamat files

Select the tiamat *.dnajson* file to import, making sure that *tiamat* is selected as file format. Binary *.dna* Tiamat files need to be reopened in Tiamat and saved to the text-based *.dnajson*. Select Tiamat version (1 or 2), then select nucleic acit type (DNA or RNA).

By default, nucleotides without assigned base types will be given a random type. However, it is also possible to select a fixed default base.

Importing PDB files

While I did create include the DNA PDB to oxDNA converter from Tacoxdna in *tacoxdna.js*, a more versatile PDB import was created by Jonah Procyk to support his ANM-oxDNA model [47]. Simply drag and drop (or load) a PDB file into an oxView window and the DNA, RNA and/or protein it contains will be automatically converted and loaded.

6.1.2 Multi-component designs

Designs spread across multiple files (or even multiple design tools) can be easily combined in oxView by simply importing them all and using the editing tools to arrange and connect them properly.

6.1.3 Far-from-physical caDNAno designs

Some structures, while converted without failure to the oxDNA format, will have a very far-from-physical configuration due to the way they are drawn in caDNAno. As mentioned above, since it is only possible to draw all helices parallel to each other, on a lattice, backbone bonds may be very elongated, creating high energies and/or topological problems.

The oxDNA software already includes relaxation procedures for such structures, bringing them together in a slow and controlled manner using a specified maximum backbone force. However, for large structures, this can take a very long time, even while using GPU simulation.

In such cases, another software I have investigated, called mrdna (Multi-resolution DNA nanotechnology), proves very useful[48]. It is a python package using ARBD (Atomic Resolution Brownian Dynamics), both being developed by Chris Maffeo, to simulate double- and single-stranded DNA structures at multiple levels of coarse-graining. Since these simulations are more coarse-grained than oxDNA, they run significantly faster. Furthermore, after I have been in contact with Chris, mrdna can now also be configured to output and simulate oxDNA

configurations as the final simulation step. Because of this, mrdna can also be useful for converting simple structures if tacoxDNA fails to parse the caDNAno file.

6.2 Rigid-body manipulation and dynamics

Rapid relaxation of converted cadnano designs

Rigid-body manipulation [30]. The translation and rotation tools in oxView allow users to select and rearrange blocks of nucleotides as rigid bodies.

Dynamics [49]

Manually, on import or through DBSCAN algorithm [50].

Clusters held together with spring forces at each shared backbone bond, with a magnitude of

$$f_{\text{spr}} = c_{spr}(l - l_r),$$

where c_{spr} is a spring constant, l is the current bond length and l_r is the relaxed bond length. To avoid overlaps, a simple linear repulsive force, of magnitude

$$f_{\text{rep}} = max\left(c_{rep}\left(1 - \frac{d}{r_a + r_b}\right), 0\right)$$

is added between the centre of each group, where c_{rep} is a repulsion constant, d is the distance between the two centres of mass, and $r_a + r_b$ is the sum of the group radii (the greatest distance they can be while still overlapping).

6.3 Editing designs

6.4 Example conversions

A selection of the DNA designs I have relaxed are shown in Figure 5.10. The first two examples, adopted from [38] and [39] and illustrated in Figure 5.10.a and 5.10.b, are both quite straightforward to relax in oxDNA, although the relaxation is much faster using mrdna.

The tensegrity kite structure, adopted from [40] is harder to relax since, as seen in the first image in Figure 5.10.c, the two helix bundles are drawn parallel

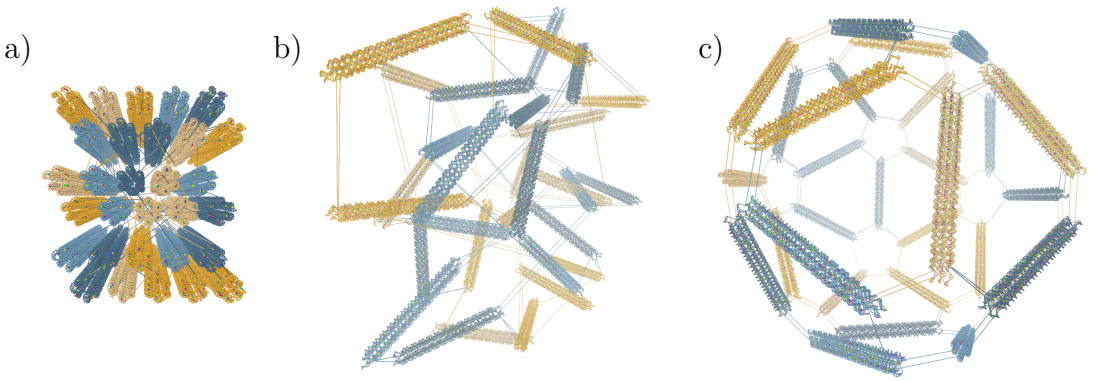

Figure 6.1: Rigid-body dynamics of clusters. Snapshots from the automatic rigid-body relaxation of an icosahedron, starting with the configuration converted from caDNAno **a)**, through the intermediate **b)** where the dynamics are applied, and **c)** the final resulting relaxed state.

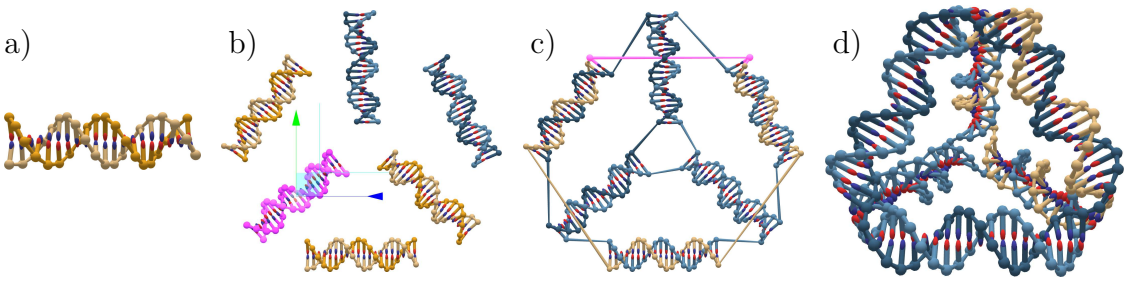

Figure 6.2: Designing the DNA tetrahedron from [51] using the oxView editing tools. a) The initial helix created. b) Duplicated helices being translated into place. c) Strands ligated together. d) The resulting 3D tetrahedron shape, as seen after applying rigid-body dynamics.

to each other in caDNAno. Given enough time to relax, they should still become orthogonal, but a much more efficient way is to write a mrdna script to rotate one helix bundle so that it is orthogonal from the start, as seen in the middle image of 5.10.c. The remaining overstretched bonds are then quickly relaxed using mrdna.

Finally, the Möbius strip, adopted from [41], is particularly tricky to relax, since the caDNAno design have all helices drawn in the same plane, with bonds from each end stretching through the whole structure and intersecting at a single point, as can be seen in the first image of Figure 5.10.d. With some help from Chris Maffeo, however, I was able to use a mrdna script to edit the structure into a configuration much easier to relax, as seen in the second image of Figure 5.10.d. Since the caDNAno design does not make it clear if the Möbius strip

Tool	Description
	**Create** a new strand from a given sequence. Select *duplex mode* to instead create a helix.
	**Copy** the selected elements (Ctrl+C).
	**Cut** the selected elements (Ctrl+X).
	**Paste** elements from clipboard (Ctrl+V to paste in original position, or Ctrl+Shift+V to paste in front of camera).
	**Delete** all currently selected elements. (delete)
	**Ligate** two strands by selecting the 3' and 5' endpoint elements to connect (L)
	**Nick** a strand at the selected element (N)
	**Extend** strand from the selected element with the given sequence. Select *duplex mode* to also extend the complementary strand.
	**Insert** (add) elements within a strand after the selected element
	**Skip** (remove) selected elements within a strand.
	**Rotate** selected elements around their center of mass (R).
	**Translate** currently selected elements (T).
	**Move to**. Move other selected elements to the position of the most recently selected element.
	**Connect 3' duplex**. Connects the 3' ends of two selected staple strands with a duplex, generated from the sequence input.
	**Connect 5' duplex**. Connects the 5' ends of two selected staple strands with a duplex, generated from the sequence input.
	**Set** the sequence of currently selected elements. Select duplex mode to also set the complementary sequence on paired elements.
	**Get**. Assigns the sequence of selected bases to the sequence input.
	**Reverse complement**. Generates the reverse complement of a provided sequence.
	**Search**. Highlights the position the provided sequence in each strand, if present.

Table 6.1: Editing tools available in oxView

should be left-handed or right-handed, this is also decided in the script; changing the rotational direction will produce a mirrored version of the structure, as seen in the third image of Figure 5.10.d.

6.5 Visualization options

Depending on the design, a user of oxView might want to modify the visualization settings to make certain features of the structure more or less visible.

Centring with periodic boundary conditions. A useful utility when visualizing an oxDNA trajectory is to keep the structure centred at the origin, stopping it from drifting out of view. The method described in [52] was used to achieve centring while taking periodic boundary conditions into account.

In essence, for each coordinate $p_j = \left(p_x^j, p_y^j, p_z^j\right)$ in the centring set of size n, each dimension $i \in \{x, y, z\}$ gets averaged in its own variable α_i, representing its 1D interval as a 2D circle (where the circumference b_i is the bounding box side length:

$$c_i = \frac{1}{n} \sum_{j=1}^{n} \left[\cos\left(\alpha_i^j\right), \sin\left(\alpha_i^j\right)\right]$$

Where $\alpha_i^j = \frac{2\pi}{b_i} p_i^j$ is the angle on the unit circle and c_i is the average 2D position representing dimension i. Finally, the averages are converted back to cartesian coordinates:

$$cm_i = \pi + \frac{b_i}{2\pi} atan2(-c_{i,y}, -c_{i,x})$$

Change component sizes , change colours, fog, virtual reality.

6.6 Exporting designs

Once a structure has been created in or loaded into oxView, it can be exported in a variety of formats.

6.6.1 Exporting oxDNA simulation files

Also force files

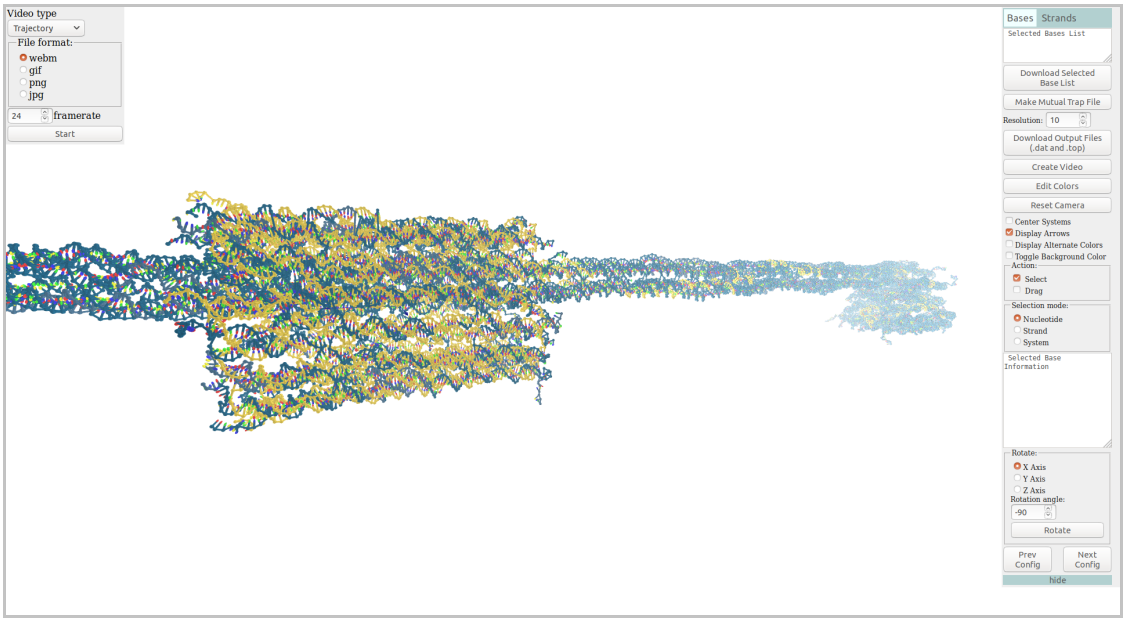

Figure 6.3: Screenshot of the online oxView tool, exporting a video of a slider on a rail from a simulated trajectory.

6.6.2 Exporting other 3D formats

6.6.3 Exporting sequence files

6.6.4 Saving image files

6.6.5 Creating videos

Trajectory

Leminiscate

6.7 Converting RNA origami designs

During my secondment at the Andersen lab in Aarhus, I worked with converting RNA structures designed using their ASCII-based blueprint format into oxRNA simulation files. Examples of converted structures are shown in Figure 6.4. The Andersen lab already has scripts, soon to be published, for parsing their blueprint files and building the corresponding PDB structures (the first two columns of Figure 6.4). However, the resulting PDB files are not relaxed and would take a long time to relax using all-atom simulation. As such, I modified the tacoxDNA [43] PDB parser to enable PDB-to-oxRNA conversion. I have contacted Lorenzo to let him

know about these additions and hopefully, they will be included in a future version of tacoxDNA. One issue I noticed with the conversion was that the asterisk (*) and prime (') characters were used interchangeably in the RNA motif library to name atoms of the sugar group, causing problems with the tacoxDNA script, which only recognizes the prime. This has already been fixed in tacoxDNA. The third column of Figure 6.4 shows the structures relaxed and simulated in oxRNA, some significantly different from the previously available PDB models in the second column.

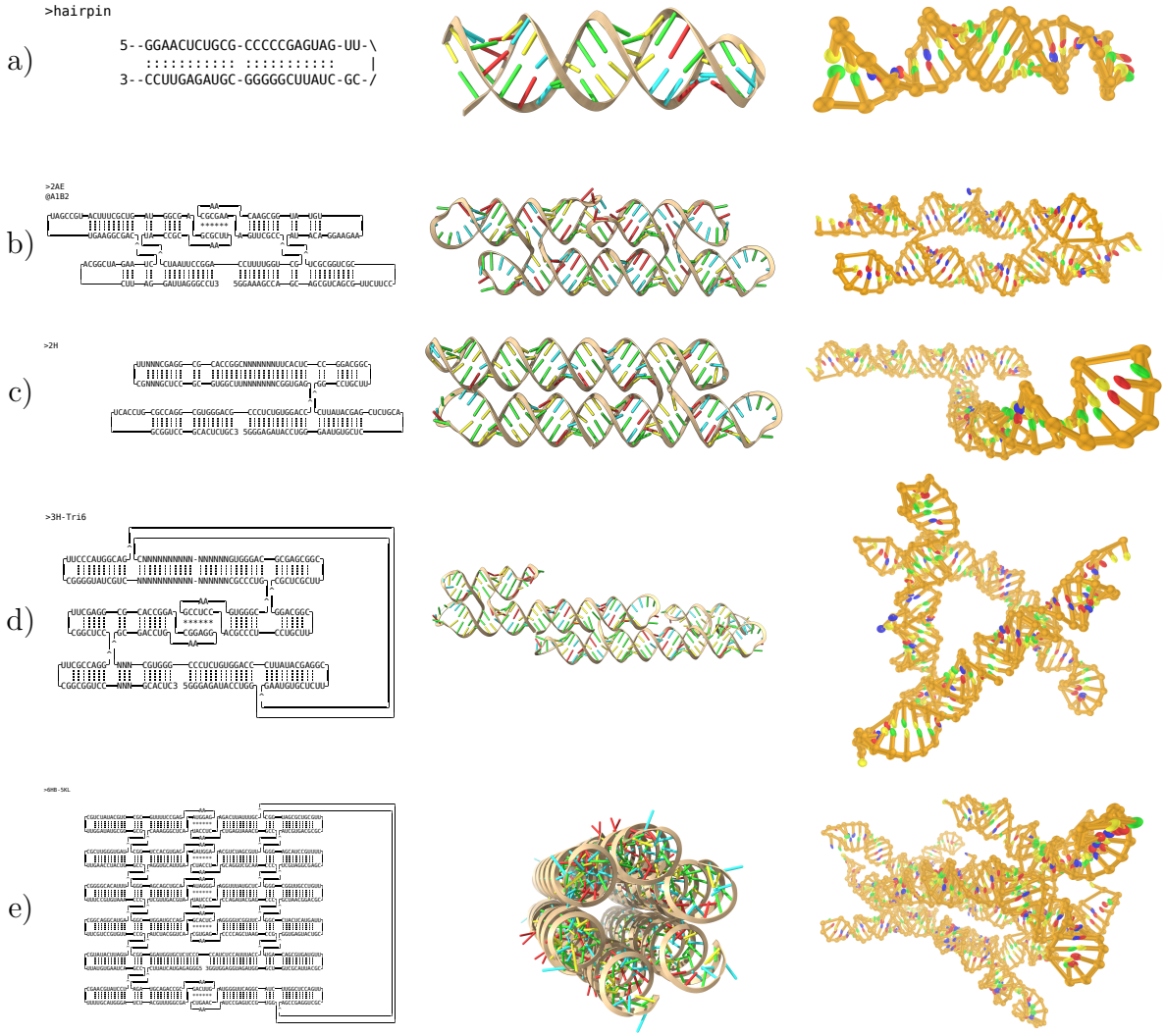

Figure 6.4: Conversion and simulation of various RNA designs. Each row, from left to right, shows the ASCII blueprint design, the PDB model (visualised using ChimeraX), and a frame from the simulated structure (visualised using oxView). **a)** Is a simple hairpin loop. **b)** is a two-helix bundle tile used in [7]. **c)** is two helices connected by a double crossover, analysing the flexibility of such a motif. **d)** is a possible design for a tensegrity triangle. **e)** is a siz-helix bundle.

You live and learn. At any rate, you live.

— Douglas Adams, Mostly Harmless

7 Conclusion

7.1 Individual module design

7.2 Modular assembly

7.3 Future work

Appendices

A

Possible appendix chapter

Appendices are just like chapters. Their sections and subsections get numbered and included in the table of contents; figures and equations and tables added up, etc. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed et dui sem. Aliquam dictum et ante ut semper. Donec sollicitudin sed quam at aliquet. Sed maximus diam elementum justo auctor, eget volutpat elit eleifend. Curabitur hendrerit ligula in erat feugiat, at rutrum risus suscipit. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Integer risus nulla, facilisis eget lacinia a, pretium mattis metus. Vestibulum aliquam varius ligula nec consectetur. Maecenas ac ipsum odio. Cras ac elit consequat, eleifend ipsum sodales, euismod nunc. Nam vitae tempor enim, sit amet eleifend nisi. Etiam at erat vel neque consequat.

References

[1] Chris R Calladine and Horace Drew. *Understanding DNA: the molecule and how it works*. Academic press, 1997.

[2] Nadrian C. Seeman. *Structural DNA Nanotechnology*. Cambridge University Press, 2016.

[3] Paul WK Rothemund. "Folding DNA to create nanoscale shapes and patterns". In: *Nature* 440.7082 (2006), p. 297.

[4] Thomas E Ouldridge, Ard A Louis, and Jonathan PK Doye. "DNA nanotweezers studied with a coarse-grained model of DNA". In: *Physical review letters* 104.17 (2010), p. 178101.

[5] Shawn M Douglas et al. "Rapid prototyping of 3D DNA-origami shapes with caDNAno". In: *Nucleic acids research* 37.15 (2009), pp. 5001–5006.

[6] Peixuan Guo. "The emerging field of RNA nanotechnology". In: *Nature nanotechnology* 5.12 (2010), p. 833.

[7] Cody Geary, Paul WK Rothemund, and Ebbe S Andersen. "A single-stranded architecture for cotranscriptional folding of RNA nanostructures". In: *Science* 345.6198 (2014), pp. 799–804.

[8] Steffen L Sparvath, Cody W Geary, and Ebbe S Andersen. "Computer-aided design of RNA origami structures". In: *3D DNA Nanostructure*. Springer, 2017, pp. 51–80.

[9] Cody Geary et al. "RNA origami design tools enable cotranscriptional folding of kilobase-sized nanoscaffolds". In: *Nature chemistry* 13.6 (2021), pp. 549–558.

[10] Nadrian C Seeman. "Nucleic acid junctions and lattices". In: *Journal of theoretical biology* 99.2 (1982), pp. 237–247.

[11] Erik Winfree et al. "Design and self-assembly of two-dimensional DNA crystals". In: *Nature* 394.6693 (1998), p. 539.

[12] Luvena L Ong et al. "Programmable self-assembly of three-dimensional nanostructures from 10,000 unique components". In: *Nature* 552.7683 (2017), pp. 72–77.

[13] Grigory Tikhomirov, Philip Petersen, and Lulu Qian. "Fractal assembly of micrometre-scale DNA origami arrays with arbitrary patterns". In: *Nature* 552.7683 (2017), p. 67.

[14] Klaus F Wagenbauer, Christian Sigl, and Hendrik Dietz. "Gigadalton-scale shape-programmable DNA assemblies". In: *Nature* 552.7683 (2017), p. 78.

[15] Christian Sigl et al. "Programmable icosahedral shell system for virus trapping". In: *Nature Materials* 20.9 (2021), pp. 1281–1289.

[16] Zhiwei Lin et al. “Engineering Organization of DNA Nano-Chambers through Dimensionally Controlled and Multi-Sequence Encoded Differentiated Bonds”. In: *Journal of the American Chemical Society* 142.41 (2020). PMID: 32902966, pp. 17531–17542. eprint: https://doi.org/10.1021/jacs.0c07263. URL: https://doi.org/10.1021/jacs.0c07263.

[17] Mingyang Wang et al. “Programmable Assembly of Nano-architectures through Designing Anisotropic DNA Origami Patches”. In: *Angewandte Chemie International Edition* 59.16 (2020), pp. 6389–6396. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/anie.201913958. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/anie.201913958.

[18] Hao Wang. “Proving theorems by pattern recognition—II”. In: *Bell system technical journal* 40.1 (1961), pp. 1–41.

[19] Erik Winfree. “Algorithmic self-assembly of DNA”. PhD thesis. California Institute of Technology, 1998.

[20] SE Ahnert et al. “Self-assembly, modularity, and physical complexity”. In: *Physical Review E* 82.2 (2010), p. 026117.

[21] Iain G Johnston et al. “Evolutionary dynamics in a simple model of self-assembly”. In: *Physical Review E* 83.6 (2011), p. 066105.

[22] Ming Li and P. M. B. Vitányi. *An introduction to Kolmogorov complexity and its applications [electronic resource]*. eng. Fourth edition. Texts in computer science. Cham, 2019.

[23] Kamaludin Dingle, Chico Q Camargo, and Ard A Louis. “Input–output maps are strongly biased towards simple outputs”. In: *Nature communications* 9.1 (2018), p. 761.

[24] Kamaludin Dingle, Guillermo Valle Pérez, and Ard A Louis. “Generic predictions of output probability based on complexities of inputs and outputs”. In: *Scientific reports* 10.1 (2020), pp. 1–9.

[25] Flavio Romano et al. “Designing patchy interactions to self-assemble arbitrary structures”. In: *Physical Review Letters* 125.11 (2020), p. 118003.

[26] Shawn M. Douglas et al. “Rapid prototyping of 3D DNA-origami shapes with caDNAno”. In: *Nucleic Acids Research* 37.15 (2009), pp. 5001–5006.

[27] David Doty, Benjamin L Lee, and Tristan Stérin. “scadnano: A browser-based, scriptable tool for designing DNA nanostructures”. In: *DNA 2020: Proceedings of the 26th International Meeting on DNA Computing and Molecular Programming*. Ed. by Cody Geary and Matthew J. Patitz. Vol. 174. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020, 9:1–9:17. URL: https://drops.dagstuhl.de/opus/volltexte/2020/12962.

[28] Nick Conway and Shawn Douglas. *Windows Installation - caDNAno*. URL: https://cadnano.org/windows-installation.html (visited on 08/20/2021).

[29] Haichao Miao et al. “Multiscale Visualization and Scale-adaptive Modification of DNA Nanostructures”. In: *IEEE Transactions on Visualization and Computer Graphics* 24.1 (Jan. 2018). URL: https://www.cg.tuwien.ac.at/research/publications/2018/miao_tvcg_2018/.

[30] Chao-Min Huang et al. “Uncertainty quantification of a DNA origami mechanism using a coarse-grained model and kinematic variance analysis”. In: *Nanoscale* 11.4 (2019), pp. 1647–1660.

[31] James C Phillips et al. “Scalable molecular dynamics with NAMD”. In: *Journal of computational chemistry* 26.16 (2005), pp. 1781–1802.

[32] Wendy D Cornell et al. “A second generation force field for the simulation of proteins, nucleic acids, and organic molecules J. Am. Chem. Soc. 1995, 117, 5179-5197”. In: *Journal of the American Chemical Society* 118.9 (1996), pp. 2309–2309.

[33] Bernard R Brooks et al. “CHARMM: a program for macromolecular energy, minimization, and dynamics calculations”. In: *Journal of computational chemistry* 4.2 (1983), pp. 187–217.

[34] Jejoong Yoo and Aleksei Aksimentiev. “In situ structure and dynamics of DNA origami determined through molecular dynamics simulations”. In: *Proceedings of the National Academy of Sciences* 110.50 (2013), pp. 20099–20104.

[35] Aditya Sengar et al. “A primer on the oxDNA model of DNA: When to use it, how to simulate it and how to interpret the results”. In: *arXiv preprint arXiv:2104.11567* (2021).

[36] Rahul Sharma et al. “Characterizing the motion of jointed DNA nanostructures using a coarse-grained model”. In: *ACS nano* 11.12 (2017), pp. 12426–12435.

[37] Petr Šulc et al. “A nucleotide-level coarse-grained model of RNA”. In: *The Journal of chemical physics* 140.23 (2014), 06B614_1.

[38] Thomas Gerling et al. “Dynamic DNA devices and assemblies formed by shape-complementary, non–base pairing 3D components”. In: *Science* 347.6229 (2015), pp. 1446–1452.

[39] Reza M Zadegan et al. “Construction of a 4 zeptoliters switchable 3D DNA box origami”. In: *ACS nano* 6.11 (2012), pp. 10050–10053.

[40] Tim Liedl et al. “Self-assembly of three-dimensional prestressed tensegrity structures from DNA”. In: *Nature nanotechnology* 5.7 (2010), p. 520.

[41] Dongran Han et al. “Folding and cutting DNA into reconfigurable topological nanostructures”. In: *Nature nanotechnology* 5.10 (2010), p. 712.

[42] Do-Nyun Kim et al. “Quantitative prediction of 3D solution shape and flexibility of nucleic acid nanostructures”. In: *Nucleic acids research* 40.7 (2012), pp. 2862–2868.

[43] Antonio Suma et al. “TacoxDNA: A user-friendly web server for simulations of complex DNA structures, from single strands to origami”. In: *Journal of Computational Chemistry* (2019). URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/jcc.26029.

[44] Erik Poppleton et al. “Design, optimization and analysis of large DNA and RNA nanostructures through interactive visualization, editing and molecular simulation”. In: *Nucleic acids research* 48.12 (2020), e72–e72.

[45] Antonio Suma et al. “TacoxDNA: A user-friendly web server for simulations of complex DNA structures, from single strands to origami”. In: *Journal of computational chemistry* 40.29 (2019), pp. 2586–2595.

[46] Erik Benson et al. “DNA rendering of polyhedral meshes at the nanoscale”. In: *Nature* 523.7561 (2015), pp. 441–444.

[47] Jonah Procyk, Erik Poppleton, and Petr Šulc. “Coarse-grained nucleic acid–protein model for hybrid nanotechnology”. In: *Soft Matter* 17.13 (2021), pp. 3586–3593.

[48] Chris Maffeo and Aleksei Aksimentiev. *Multi-resolution simulations of self-assembled DNA nanostructures*. 2018. URL: https://gitlab.engr.illinois.edu/tbgl/tutorials/multi-resolution-dna-nanotechnology/tree/master.

[49] David Baraff. *An introduction to physically based modeling: Rigid Body Simulation I — Unconstrained Rigid Body Dynamics*. 1997.

[50] Martin Ester et al. “A density-based algorithm for discovering clusters in large spatial databases with noise.” In: *Kdd*. Vol. 96. 34. 1996, pp. 226–231.

[51] Russell P Goodman et al. “Rapid chiral assembly of rigid DNA building blocks for molecular nanofabrication”. In: *Science* 310.5754 (2005), pp. 1661–1665.

[52] Linge Bai and David Breen. “Calculating Center of Mass in an Unbounded 2D Environment”. In: *Journal of Graphics Tools* 13.4 (2008), pp. 53–60. eprint: https://doi.org/10.1080/2151237X.2008.10129266. URL: https://doi.org/10.1080/2151237X.2008.10129266.