Self-Assembly of Arbitrary Shapes Using RNAse Enzymes: Meeting the Kolmogorov Bound with Small Scale Factor

Erik Demaine, Matthew Patitz, Robert Schweller, Scott Summers

Self-Assembly of Arbitrary Shapes Using RNAse Enzymes: Meeting the Kolmogorov Bound with Small Scale Factor (extended abstract)*

Erik D. Demaine[1], Matthew J. Patitz[2], Robert T. Schweller[3], and Scott M. Summers[4]

1 **MIT Computer Science and Artificial Intelligence Laboratory, 32 Vassar St., Cambridge, MA 02139, USA.**
edemaine@mit.edu

2 **Department of Computer Science, University of Texas–Pan American, Edinburg, TX 78539, USA.**
mpatitz@cs.panam.edu

3 **Department of Computer Science, University of Texas–Pan American, Edinburg, TX 78539, USA.**
schwellerr@cs.panam.edu

4 **Department of Computer Science and Software Engineering, University of Wisconsin–Platteville, Platteville, WI 53818, USA.**
summerss@uwplatt.edu

Abstract

We consider a model of algorithmic self-assembly of geometric shapes out of square Wang tiles studied in SODA 2010, in which there are two types of tiles (e.g., constructed out of DNA and RNA material) and one operation that destroys all tiles of a particular type (e.g., an RNAse enzyme destroys all RNA tiles). We show that a single use of this destruction operation enables much more efficient construction of arbitrary shapes. In particular, an arbitrary shape can be constructed using an asymptotically optimal number of distinct tile types (related to the shape's Kolmogorov complexity), after scaling the shape by only a logarithmic factor. By contrast, without the destruction operation, the best such result has a scale factor at least linear in the size of the shape and is connected only by a spanning tree of the scaled tiles. We also characterize a large collection of shapes that can be constructed efficiently without any scaling.

1998 ACM Subject Classification F. Theory of Computation

Keywords and phrases Biomolecular computation, RNAse enzyme self-assembly, algorithmic self-assembly, Komogorov complexity

Digital Object Identifier 10.4230/LIPIcs.STACS.2011.201

1 Introduction

DNA self-assembly research attempts to harness the power of synthetic biology to manipulate matter at the nanoscale. The general goal of this field is to design a simple system of particles (e.g., DNA strands) that efficiently assemble into a desired macroscale object. Such technology is fundamental to the field of nanotechnology and has the potential to allow for massively parallel, bottom-up fabrication of complex nanodevices, or the implementation of

* This research was supported in part by NSF grant CDI-0941538.

28th Symposium on Theoretical Aspects of Computer Science (STACS'11).
Editors: Thomas Schwentick, Christoph Dürr; pp. 201–212

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

a biological computer. Motivated by experimental DNA assemblies of basic building blocks or DNA *tiles* [5,7,14,16,17,19,26], the *tile self-assembly model* [18] has emerged as a premier theoretical model of self-assembly. Tile self-assembly models particles as four-sided Wang tiles which float randomly in the plane and stick to one another when abutting edges have sufficient affinity for attachment.

Perhaps the most fundamental question within the tile self-assembly model is how efficiently, in terms of the number of distinct tile types needed, can a target shape be uniquely assembled. For some special classes of shapes such as rectangles and squares, the problem has been considered in depth under a number of tile-based self-assembly models. More generally, researchers have considered the complexity of assembling arbitrary shapes [9,11,20]. In particular, Soloveichik and Winfree [20] show that any shape, modulo scaling, can be self-assembled with a number of tile types close to the Kolmogorov complexity of the target shape. While intriguing from a theoretical standpoint, this result has an important drawback: it assembles an arbitrarily large scaled-up version of the target shape, rather than the exact target shape. It is conceivable that a reasonable scale factor could be tolerated in practice by simply engineering smaller tiles, but the scale factors needed for the Soloveichik-Winfree construction are unbounded in general, proportional to the running time of the Kolmogorov machine that generates the shape, which is at least linear in the size of the target shape in all cases. This extreme resolution loss motivates the search for a practical model and construction that can achieve extremely small scale factors while retaining the Kolmogorov-efficient tile complexity for general shapes.

Our results. We achieve Kolmogorov-efficient tile complexity of general shapes with a logarithmic bounded scale factor, using the experimentally motivated *Staged RNA Assembly Model (SRAM)* introduced in [1]. The SRAM extends the standard tile self-assembly model by distinguishing all tile types as consisting of either DNA or RNA material. Further, in a second stage of assembly, an *RNase enzyme* may be added to the system which dissolves all RNA tiles, thus potentially breaking assemblies apart and allowing for new assemblies to form. While this modification to the model is simple and practically motivated (the idea was first mentioned in [18]), we show that the achievable scale factor for Kolmogorov-efficient assembly of general shapes drops dramatically: for arbitrary shapes of size n, a scale factor of $O(\log n)$ is achieved, and for a large class of "nice" shapes, the Kolmogorov optimal tile complexity can be achieved without scaling (scale factor 1). Refer to Figure 1. Note that the lower bound proof of [20] holds with a simple modification to the program that simulates self-assembly in the SRAM. Further, we show that arbitrarily large portions of infinite computable patterns of the plane can be weakly assembled within the SRAM. Such assembly has been proved impossible in the standard tile assembly model [13], illustrating an important distinction in the power of SRAM compared to the standard tile assembly model.

In addition to tile complexity and scale factor, we also address the metrics of *connectivity* and *addressability*. *Full connectivity* denotes whether all adjacent tiles making up the target shape share positive strength bonds, a desirable property as it creates a stable final assembly. All of our finite constructions are fully connected, unlike the previous result of [20] which just connected a spanning tree of the scaled tiles, making for a potentially very floppy construction. *Addressability* denotes whether a construction is able to assign arbitrary binary labels to the tiles that make up the final assembly. Addressability may have important practical applications for assemblies that are to serve as scaffolding for the fabrication of nanodevices such as circuits in which specific components must be attached to specific locations in the assembled shape. Our $O(\log n)$-scale construction provides the flexibility to encode an arbitrary binary label within the tile types of each scaled-up position in the

General shape S with n points	**Tile Types**	**Stages**	**Scale**	**Connectivity**
Previous work [20]	$\Theta(K(S)/\log K(S))$	1	unbounded	partial
Arbitrary shapes (Thm. 3.2)	$\Theta(K(S)/\log K(S))$	2	$O(\log n)$	full
"Nice" shapes (Thm. 4.2)	$\Theta(K(S)/\log K(S))$	2	1	full
Infinite computable pattern S	**Tile Types**	**Stages**	**Scale**	**Connectivity**
Computable patterns (Sec. 4.4)	$\Theta(K(S)/\log K(S))$	2	1	partial

Table 1 Summary of the tile complexities, stage complexities, scale factors, and connectivity of our RNA staged assembly constructions compared with relevant previous work. The value $K(S)$ denotes the Kolmogorov complexity of a given shape or pattern S, and n denotes the size of (number of points in) S.

assembled shape, thus yielding a high degree of addressability, while our 1-scale construction allows complete addressability. See [10] for a version of this paper that includes color images and a full technical appendix.

2 Preliminaries

We work in the 2-dimensional discrete space $\mathbb{Z}^2$. Let $U_2 = \{(0,1),(1,0),(0,-1),(-1,0)\}$ be the set of all unit vectors in $\mathbb{Z}^2$. We write $[X]^2$ for the set of all 2-element subsets of a set X. All *graphs* here are undirected graphs, i.e., ordered pairs $G = (V, E)$, where V is the set of *vertices* and $E \subseteq [V]^2$ is the set of *edges*. A *grid graph* is a graph $G = (V, E)$ in which $V \subseteq \mathbb{Z}^2$ and every edge $\{\vec{a}, \vec{b}\} \in E$ has the property that $\vec{a} - \vec{b} \in U_2$. The *full grid graph* on a set $V \subseteq \mathbb{Z}^2$ is the graph $G_V^{\#} = (V, E)$ in which E contains *every* $\{\vec{a}, \vec{b}\} \in [V]^2$ such that $\vec{a} - \vec{b} \in U_2$.

A *shape* is a set $S \subseteq \mathbb{Z}^2$ such that $G_S^{\#}$ is connected. In this paper, we consider scaled-up versions of finite shapes. Formally, if X is a shape and $c \in \mathbb{N}$, then a *c-scaling* of S is defined as the set $S^c = \left\{(x,y) \in \mathbb{Z}^2 \mid \left(\left\lfloor \frac{x}{c} \right\rfloor, \left\lfloor \frac{y}{c} \right\rfloor\right) \in X\right\}$. Intuitively, S^c is the shape obtained by replacing each point in S with a $c \times c$ block of points. We refer to the natural number c as the *scaling factor* or *resolution loss*. Note that scaled shapes have been studied extensively in the context of a variety of self-assembly systems [6, 9, 11, 20, 25].

Fix some universal Turing machine U. The *Kolmogorov complexity* of a shape S, denoted by $K(S)$, is the size of the smallest program π that outputs an encoding of a list of all the points in S. In other words $K(S) = \min\{|\pi| \mid U(\pi) = \langle S \rangle\}$. The reader is encouraged to consult [21] for a more detailed discussion of Kolmogorov complexity.

Here we give a sketch of a variant of Erik Winfree's abstract Tile Assembly Model (aTAM) [22, 23] known as the *two-handed* aTAM, which has been studied previously under various names [2, 4, 8, 9, 15, 24]. Please see [12] for a more detailed description of the model and our notation.

A *tile type* is a unit square with four sides, each having a *glue* consisting of a *label* (a finite string) and *strength* (0, 1, or 2). We assume a finite set T of tile types, but an infinite number of copies of each tile type, each copy referred to as a *tile*. A *supertile* (a.k.a., *assembly*) is a positioning of tiles on the integer lattice $\mathbb{Z}^2$. Two adjacent tiles in a supertile *interact* if the glues on their abutting sides are equal. Each supertile induces a *binding graph*, a grid graph whose vertices are tiles, with an edge between two tiles if they interact. The supertile is *τ-stable* if every cut of its binding graph has strength at least τ, where the weight of an edge is the strength of the glue it represents. That is, the supertile is stable if at least energy τ is required to separate the supertile into two parts. A *tile assembly system* (TAS) is a pair $\mathcal{T} = (T, \tau)$, where T is a finite tile set and τ is the *temperature*, usually 1 or 2. Throughout this paper $\tau = 2$ (unless explicitly stated otherwise). Given a TAS $\mathcal{T} = (T, \tau)$, a supertile

is *producible* if either it is a single tile from T, or it is the τ-stable result of translating two producible assemblies. A supertile α is *terminal* if for every producible supertile β, α and β cannot be τ-stably attached. A TAS is *directed* (a.k.a., *deterministic*, *confluent*) if it has only one terminal, producible supertile. Given a connected shape $X \subseteq \mathbb{Z}^2$, a TAS $\mathcal{T}$ *produces* X *uniquely* if every producible, terminal supertile places tiles only on positions in X (appropriately translated if necessary).

RNA tiles and RNAse enzyme

In this paper, we assume that each tile type is defined as being composed of either DNA or RNA. By careful selection of the actual nucleotides used to create the glues, tile types of any combination of compositions can bind together. The utility of distinguishing RNA-based tile types comes from that fact that, at prescribed points during the assembly process, the experimenter can add an RNAse enzyme to the solution which causes all tiles composed of RNA to dissolve. We assume that, when this occurs, all portions of all RNA tiles are completely dissolved, including glue portions that may be bound to DNA tiles, returning the previously bound edges of those DNA tiles to unbound states.

More formally, for a given supertile Γ that is stable at temperature τ, when the RNAse enzyme is added, all positions in Γ which are occupied by RNA tiles change to the empty tile. The resultant supertile may not be τ-stable and thus defines a multiset of subsupertiles consisting of the maximal stable supertiles of Γ at temperature τ, denoted by $BREAK_\tau(\Gamma)$.

The plausibility of this model was mentioned by Rothemund and Winfree in [18], and formalized in SODA 2010 [1] when it was combined with the idea of staged assembly [9].

Staged assembly with RNA removals

Staged assembly consists of a finite sequence of stages, modeling the actions taken by an experimenter (e.g., bioengineer). A stage assembly system specifies each stage as either a tile addition stage, in which new tile types are added to the system, or an enzyme stage, in which assembled supertiles are broken into pieces by deleting all occurrences of RNA tile types. In both cases, each stage consists of an initial set of preassembled supertiles from the previous stage, unioned with a new set of tile types in the case of a tile addition stage, or the current supertile set broken into subsupertiles (which may then be able to bind to each other) in the case of an enzyme stage. From this initial set, the output of the stage is determined by the two-handed assembly model, and the stage ends once all supertiles are terminal, meaning that no further bindings can occur. It is only at this point that the next stage can be initiated.

Complexity Measures of Tile Assembly Systems

In this paper, we are primarily concerned with measuring the "complexity" of a tile assembly system with respect to the following metrics. **Tile Complexity**: we say that the *tile complexity* (sometimes called the *program-size complexity* [18]) is the number of unique tile types of the system. **Stage Complexity**: we say that the *stage complexity* is the number of stages that a particular tile system must progress through in order to produce a terminal assembly. (We sometimes also mention the *BREAK* complexity [1], which is simply the number of *BREAK* stages.) **Scale Factor**: we say that a tile system produces a shape S with *scale factor* $c \in \mathbb{N}$ if the system uniquely produces S^c. **Connectivity**: when a tile system produces a terminal assembly in which not every adjacent edge interacts with positive strength, then we say that the system has *partial connectivity*. On the other hand, a tile assembly system achieves *full connectivity* if it only produces terminal assemblies in which every abutting edge interacts with positive strength. **Addressability**: addressability (of

the final assembly of a tile assembly system) concerns the ability of a tile system to *address* or mark each tile in the final assembly with a character drawn from $\Sigma = \{0, 1\}$. Note that addressability concerns the ability of tile systems to label certain (tiles placed at) locations in their final assembly as "black" or "nonblack."

3 The Pod Construction

3.1 Partial Connectivity Construction

As a warmup to our main result, we obtain partial connectivity:

▶ **Theorem 3.1.** *For every finite shape $S \subset \mathbb{Z}^2$, there exists a staged RNA assembly system $\mathcal{T}_S$ that uniquely produces S and moreover, $\mathcal{T}_S$ has tile complexity $O\left(\frac{K(S)}{\log K(S)}\right)$, stage complexity 2, a scale factor of $O(\log |S|)$, and has partial connectivity.*

One highlight of Theorem 3.1 is that the stage complexity of $\mathcal{T}_S$ is 2, i.e., the stages in our construction consist of the initial tile addition stage followed by a single *BREAK* stage. This is the fewest stages possible in any construction that makes use of the power of the RNAse enzyme. The remainder of this section is devoted to providing a proof sketch of Theorem 3.1.

At a high level, the construction for Theorem 3.1 works by forming a $O(\log |S|) \times O(\log |S|)$ (roughly) square block to represent each point in S. The correct positioning of blocks is ensured by encoding binary strings that are unique to each pair of adjacent edges as "teeth" on the edges of the blocks. The assembly begins with a seed, composed of RNA tile types representing a Turing machine that outputs S as a list of points. An assembly which simulates that Turing machine and then outputs definitions for each of the blocks assembles first, with all tiles being composed of RNA except for those forming the blocks, which are composed of DNA. We think of these blocks as DNA "pods" growing off of the RNA assembly. A *BREAK* operation is then performed which dissolves everything except for the DNA blocks. These blocks then combine to form the scaled version of S. Details of this construction follow. Figure 1a shows the basic design of the blocks used in this construction. Figure 1b depicts the high level structure of this construction.

The seed row consists of a row of tiles that uniquely self-assemble into a binary representation of the shortest Turing machine M that outputs the definition of a desired shape S as a list of points, and then halts. Note that we use the optimal encoding scheme of [3, 20], which implies that the tile complexity of our construction is $O\left(\frac{K(S)}{\log K(S)}\right)$.

Assembly begins with the "unpacking" phase (similar to the main construction of Soloveichik and Winfree [20]). Once this simulation completes, the top row of the assembly will consist of the list of points in the shape. Next, another Turing machine, N (charged with the task of executing the algorithm defined in Section A.4 of [10]), is simulated by the assembly.

Once N halts, the top row of the assembly will consist of a sequence of binary strings that represent the binary values to be encoded along the edges of the DNA blocks. It is these blocks that will come together in a 2-handed fashion to form the final, scaled version of S. The correct positioning of the blocks is ensured by the patterns of binary teeth as well as the glues on the corners of the blocks which ensure that only complementary corners of blocks can bind (e.g., the northeast corner of one block could bind only to the northwest corner of another). For block edges which correspond to an outer edge of the shape S, instead of binary teeth a smooth edge with 0-strength glues will be formed. Note that the seed tiles, Turing machine simulation tiles, and tiles outside of the blocks are all RNA tile types which will ultimately be dissolved by RNase enzyme in the *BREAK* stage. Following the

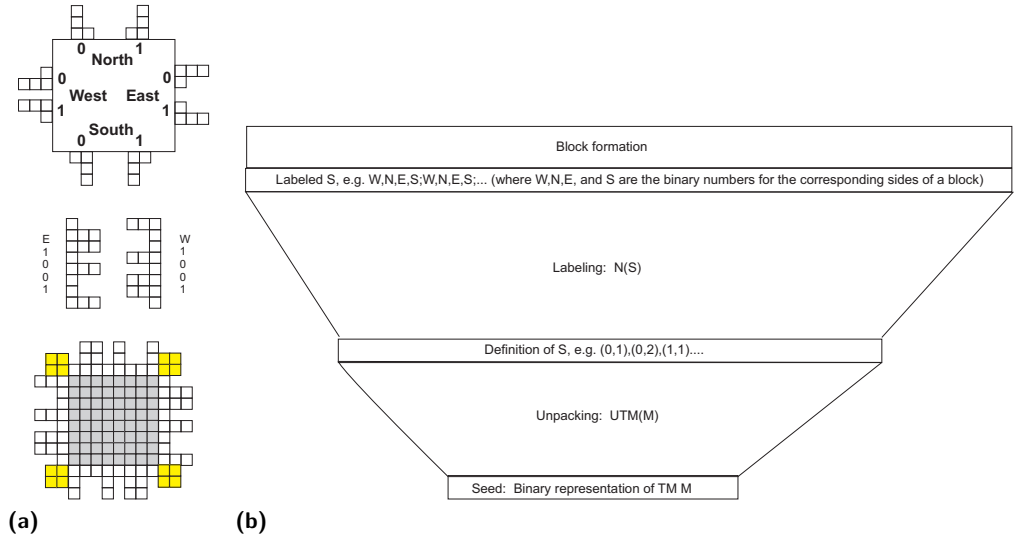

Figure 1 (a) Top: Key showing the shapes assembled for bits on each side of a block. Middle: Example East side and West side, each representing the bit pattern "1001". Bottom: Example block which has the bit pattern "1001" on each side. Note that the white tiles represent the binary patterns and have null glues on their outer edges while each exposed side of each yellow block has a single strength 1 glue exposed which is specific to its corner and direction. (b) High level overview of the main components of the pod construction.

BREAK, the $O(\log |S|) \times O(\log |S|)$ sized blocks representing each of the points in S are free to self-assemble into the scaled up version of S, thus completing the construction.

3.2 Full Addressability of Points in S

In the aTAM, tile types are allowed to have "labels" which are nonfunctional (not necessarily unique) strings associated with each tile type. Often, labels are assigned to tile types to make it easier to logically identify and group them (for instance, the "0" and "1" labels assigned to the tile types that assemble into a binary counter). In laboratory implementations of DNA tile types, tile types are often created with the equivalent of such binary labels by the inclusion or exclusion of a hairpin loop structure which projects upward above the plane of the tile, for 0 and 1 respectively (a notable example of this technique is due to Papadakis, Rothemund and Winfree [19]). This is currently done to simplify the imaging process and therefore the detection of errors that occur in the assembly. However, it is possible that in the future such projecting labels could be also used to create binding sites for additional materials, allowing the self-assembling structure to serve as a scaffolding for more complicated productions. For simplicity, we let the set of available labels be $\Sigma = \{0, 1\}$.

Here we present a construction that facilitates the arbitrary assignment of labels to subsets of locations in the final assembly. We consider such locations to be "addressable." This provides a method for associating labels, in the form of binary strings, with each of the points in S. These binary strings will be represented by rows of tiles within the blocks, each labeled with a "0" or "1."

In the construction for Theorem 3.1, it is trivial to allow the TM M encoded in the seed to also output a binary string to be used to label each/any point in S. This binary string can be passed upward through the south sides of the DNA blocks so that they are represented by the labels of the tile types which form the center of each block (either in particular, designated rows or in all rows). Of course, doing so requires an appropriate increase in tile

complexity—the *additional* complexity of encoding each string that will ultimately be printed on (e.g., used to address) each supertile in the final assembly.

This labeling method allows bit strings of length at most the width of the center portion of a DNA block (plus 2 additional tiles) to be specified for each DNA block. Only one such unique label can be specified for each block, but the row (or rows) in which it appears can be specified by M. The label can appear in any subset of the rows, or alternatively in columns. Intuitively, this is done by including a label value, which passes either upward or to the right as the center of the block assembles. At rows (or columns) that have been specified with special markers as M output the definition of the block, the label values can be "expressed" by tile types with the labels corresponding to the bit values.

3.3 Full Connectivity Construction

Recall that for the previous constructions, the only positive strength interaction between the glues of adjacent blocks occurred at the corners of those blocks. We now strengthen Theorem 3.1 as follows.

▶ **Theorem 3.2.** *For every finite shape $S \subset \mathbb{Z}^2$, there exists a staged RNA assembly system $\mathcal{T}_S$ that uniquely produces X and moreover, $\mathcal{T}_S$ has tile complexity $O\left(\frac{K(S)}{\log K(S)}\right)$, stage complexity 2, a scale factor of $O(\log |S|)$, and achieves full connectivity of the terminal assembly.*

A proof sketch of Theorem 3.2 follows. In order to generate shapes with full connectivity, the scheme proposed below requires that the scaling factor be doubled from the construction of Theorem 3.1 and also that, when the RNase enzyme is added, *there are no remaining singleton tiles (neither DNA nor RNA) in the solution*, only the terminally produced assemblies. The latter requirement is due to the fact that the teeth of the blocks produced have single strength glues all along their edges to which single tiles of the correct types could attach and prevent the proper connection of blocks. However, it is easy to remove this assumption by doubling the system temperature from $\tau = 2$ to $\tau = 4$, *and* doubling the strength of every glue that is *internal* to each DNA block while maintaining single strength glues that are on the outside of the block. Note that this additional assumption is not needed for the construction for Theorem 3.1 since with those blocks, there are no locations on the exposed sides to which singleton tiles could attach, only the correct and fully formed complementary blocks.

Figure 2 shows the procedure by which the values for the edges of a block are moved into the necessary positions relative to the edges of the block to be formed. It also shows how those values are turned into "casts" formed of RNA tiles. The high level idea is that first, before any DNA tiles can attach to the assembly, RNA tiles form a "cast" whose shape is the complement of the teeth of the block. Once the self-assembly of the portion of the cast for an edge is completed, the assembly of the DNA teeth for that side is allowed to proceed. The cast is formed as a one-tile-wide path of tiles whose order of growth is generally clockwise. Once the self-assembly of the entire cast is completed, the DNA tiles can fully form the block. Every DNA tile has strength-1 glues on every edge and attaches with its south and west sides as input sides, generally forming the block from the bottom left to the top right (more details of the cast formation can be found

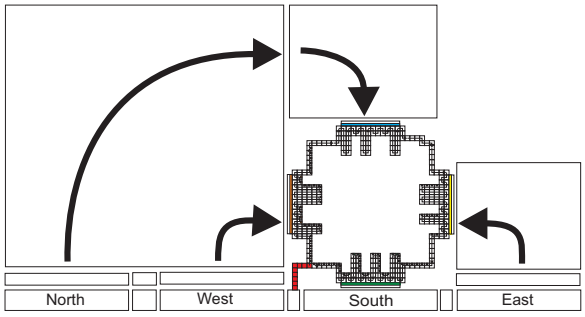

Figure 2 Positioning of block edge information.

in Section A.7 of [10]). Once the blocks form, the remainder of the construction proceeds similarly to the prior construction.

4 Self-Assembly of Shapes without Scaling

4.1 A Bounded Rectangle Decomposition of an Arbitrary Shape

We will now show how the pod construction of Section 3 can be modified to reduce the scale factor from $O(\log |S|)$ to 1 for a large class of finite shapes, while still obtaining asymptotically optimal tile complexity (according to the Kolmogorov complexity of the target shape), using just a single *BREAK* stage, and maintaining full connectivity of the final assembly. The large class of shapes will be the set of shapes that have a "bounded rectangle decomposition"–the definition of which follows.

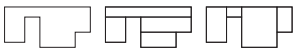

Figure 3 A shape to be formed (left) and the possible rectangle decompositions thereof (middle,right).

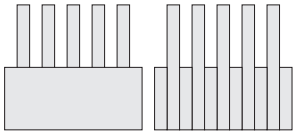

Figure 4 One possible decomposition (among several) of a shape into rectangles.

The leftmost image in Figure 3 shows an example of a simple target shape to be assembled. The middle and rightmost images show two different possible rectangle decompositions of that shape. Instead of having binary teeth along the full edges of each constituent rectangle, binary teeth need only be present at the locations where rectangles must come together, i.e., at the *interface* between two rectangles. The remainder of the outside edges can be made smooth, with 0-strength glues. Throughout this section, S denotes an arbitrary finite shape.

A shape R is a rectangle if $R = \{(x, y) \in \mathbb{Z}^2 \mid a \leq x < m + a \text{ and } b \leq y < n + b \text{ for some } a, b, m, n \in \mathbb{N}\}$. In this case, we say that R is a rectangle of width m and height n positioned at (a, b). We say that $\mathcal{R}(S) = \{R_i\}_{i=0}^{k}$, for some $k \in \mathbb{N}$ is a *rectangle decomposition* of S if for all $0 \leq i < k$, R_i is a non-empty rectangle, $\bigcup_{i=0}^{k-1} R_i = S$ and for all $i, j \in \mathbb{N}$ such that $i \neq j$, $R_i \cap R_j = \varnothing$. See Figure 3 for examples. Let $\mathcal{R} = \{R_i\}_{i=0}^{k-1}$ be a rectangle decomposition of S and suppose that R_i and R_j are rectangles in $\mathcal{R}$. For each $\vec{u} \in U_2 = \{(0, 1), (1, 0), (0, -1), (-1, 0)\}$, denote as $I^{\vec{u}}(R_i, R_j)$ the *interface between rectangles* R_i *and* R_j *in direction* $\vec{u}$, i.e., $I^{\vec{u}}(R_i, R_j)$ is the set of all points $(x, y) \in R_j$ such that $(x, y) = (w, z) + \vec{u}$ for some $(w, z) \in R_i$. It is easy to see that, for any rectangle decomposition $\mathcal{R}$, $I^{\vec{u}}(R_i, R_j)$ is the unique interface in direction $\vec{u}$ between R_i and R_j *or* $I^{\vec{u}}(R_i, R_j) = \varnothing$. For each $\vec{u} \in U_2$, the *length* of an interface $I^{\vec{u}}(R_i, R_j)$ is $|I(R_i, R_j)|$. For each $\vec{u} \in U_2$, we say that the *orientation of an interface* $I^{\vec{u}}(R_i, R_j)$ is *horizontal* if $\vec{u} \in \{(1, 0), (-1, 0)\}$ and *vertical* if $\vec{u} \in \{(0, 1), (0, -1)\}$. We say that R_i and R_j are *adjacent* if $I^{\vec{u}}(R_i, R_j) \neq \varnothing$ for some $\vec{u} \in U_2$.

▶ **Definition 4.1.** Let $\mathcal{R} = \{R_i\}_{i=0}^{k-1}$ be a rectangle decomposition of S. We say that $\mathcal{R}$ is a *bounded rectangle decomposition* if: (1) for each $l \in \mathbb{N}$,

$$\left|\left\{ \left|I^{\vec{u}}(R_i, R_j)\right| = l \;\middle|\; \vec{u} \in U_2, i, j \in \mathbb{N} \text{ and } R_i, R_j \in \mathcal{R} \right\}\right| \leq 2^{\left\lfloor \frac{l-12}{4} \right\rfloor}$$

and (2) for all $R_i, R_j \in \mathcal{R}$, if R_i and R_j are adjacent (in some particular direction $\vec{u} \in U_2$), then $\left|I^{\vec{u}}(R_i, R_j)\right| \geq 16$.

Definition 4.1 is motivated by the way we will ultimately construct tile interfaces between DNA supertiles in our forth-coming construction (discussed in the next subsection): each

supertile-supertile interface of length l can play host to at most $\left\lfloor \frac{l-12}{4} \right\rfloor$ binary "teeth" since we will use 6 tiles for each corner piece and 4 tiles for the representation of each bit in the interface. Intuitively, the first condition in Definition 4.1 says that there cannot be "too many" (i.e., roughly exponentially-many) interfaces of each length in $\mathcal{R}$, whereas the second condition is merely saying that every non-empty interface must be at least a certain length. In order to bypass the limitation imposed by the first condition, the shape could simply be scaled, with a worst possible case being a scale factor of $O(\log |S|)$.

4.2 Self-Assembly of Rectangles of Arbitrary Dimension

The construction for Theorem 3.2 can be modified to prove the following result.

▶ **Theorem 4.2.** *For every finite shape $S \subset \mathbb{Z}^2$, if S has a bounded rectangle decomposition, then there exists a staged RNA assembly system $\mathcal{T}_S$ that uniquely produces S, $\mathcal{T}_S$ has tile complexity $O\left(\frac{K(S)}{\log K(S)}\right)$, utilizes 2 stages with a single BREAK step and achieves full connectivity of the terminal assembly.*

4.3 Full Addressability of Every Tile in the Final Assembly

In this section, we sketch a construction utilizing a single *BREAK* step that assembles shapes (that can be "nicely" decomposed into rectangles) with no scaling, full connectivity, and full addressability (in the form of specifying either a 0 or 1 label to appear in every single tile position of the final assembly). This strengthens Theorem 4.2 with respect to addressability but with an additional increase in tile complexity of $O(K(B))$, where $B \subseteq S$ is the set of points to be addressed, i.e., the set of points in the final assembly at which tiles labeled with a "1" are placed, as well as requiring an additional constraint on the rectangles contained within the rectangle decomposition. For this construction, we require that there is some constant $k \in \mathbb{Z}^+$ that bounds at least one dimension of every rectangle in every valid rectangle decomposition. That is, every rectangle, although potentially arbitrarily long (or wide) in one dimension, must be no longer or wider *in the other dimension*, than k tiles.

The details of how the rectangular blocks for this construction are formed are depicted in Figure 5. Our construction can be thought of to proceed in four logical phases: the unpacking process, self-assembly of the RNA cast, self-assembly of the rectangular supertiles, and self-assembly of the target shape. The main difference with the previous construction is in the complexity of the cast and the order of assembly of the tiles forming the rectangular supertiles. At a high level, this is due to the fact that information about the specific labels, and therefore tile types–that need to eventually occupy every single position–must be propagated from the casts into the forming rectangular supertiles. This forces the constraint on one dimension of each rectangle, and the fact that the construction retains full connectivity forces the positioning of the glues on the cast that propagate the information to be greatly complicated. Details of this construction can be found in Section A.8 of [10], and a high level schematic can be seen in Figure 6.

4.4 Weak Self-Assembly of Computable Patterns

Weak self-assembly is a general notion of self-assembly that applies to the self-assembly of patterns that are in some sense "painted" on a canvas of tiles that strictly contains S (as opposed to *strict self-assembly*, which pertains to the self-assembly of a given target shape and nothing else). Intuitively, we say that a pattern $S \subseteq \mathbb{Z}^2$ weakly self-assembles if there

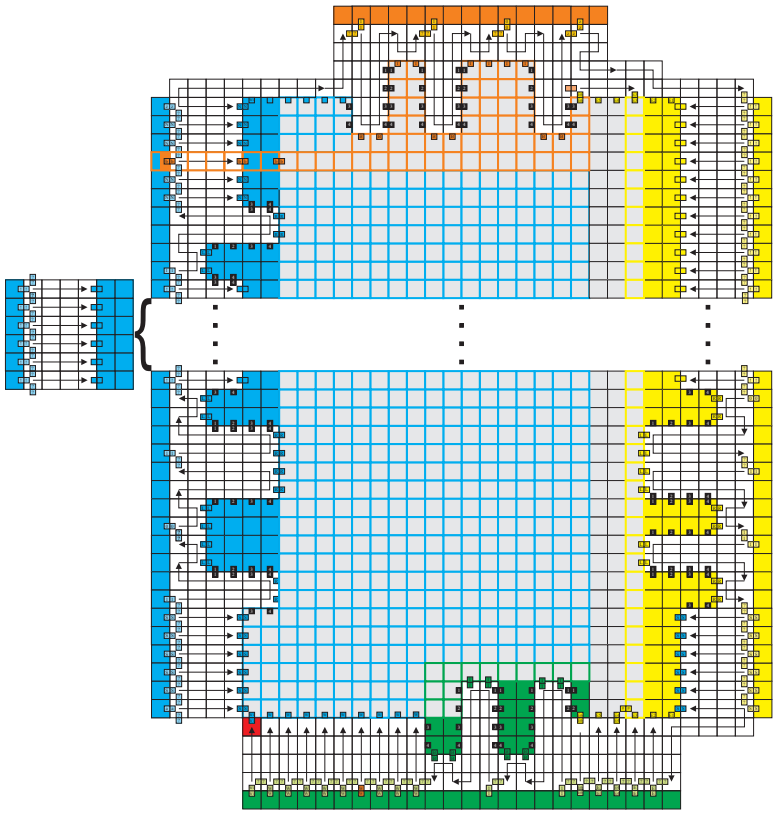

Figure 5 Schematic of the self-assembly of a fixed-width, fully addressable rectangle that will ultimately (after the RNAse enzyme is applied) participate in the self-assembly of a fully connected and fully addressable unique terminal assembly (see Section A.8 of [10] for more details). The cast forms as a single path around the entire perimeter, beginning at the bottom left side. Shaded/colored tiles are DNA tiles while white tiles are RNA. Only colored, non-grey tiles are allowed to assemble before the entire cast assembles. We depict single strength bonds as little colored (and labeled) squares along the edges of tiles. Arrows represent double strength bonds between contiguous groups of tiles through which they pass.

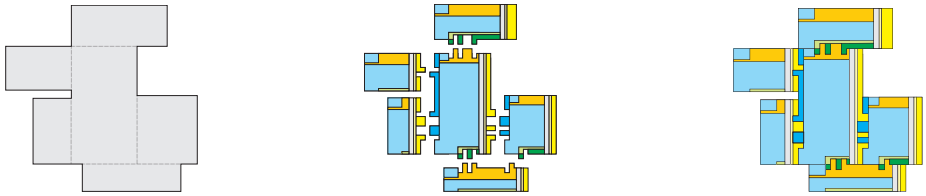

Figure 6 Each individual supertile is colored so as to correspond to the more detailed Figure 5. Note that the bottommost supertile attaches via two north-facing interfaces! (a) An example target shape X and a candidate bounded-rectangle decomposition (b) The corresponding supertiles. For the sake of example, assume the width of each supertile must not exceed that of the bottommost supertile (c) The final assembly

is a tile system that places special "black" marker tiles on—and only on—every point that belongs to the set S.

Our final construction self-assembles an arbitrarily "large" (square) portion of any computable pattern, with the size of the portion of the pattern determined simply by how long the self-assembly is allowed to proceed before the *BREAK* operation is performed. This

clearly demonstrates the fact that staged self-assembly with DNA removals is strictly more powerful than the aTAM, in terms of the weak self-assembly of patterns, as it was shown in [13] that there are (decidable) patterns that cannot weakly self-assembles in the aTAM.

Essentially, this assembly simulates a Turing machine using RNA tiles and creates pods for DNA tile rectangles with constant width (or height) and increasingly large height (or width). Tiles on these pods are labeled corresponding to the portion of the pattern which they will occupy. Figure 7 demonstrates the manner in which these rectangles will ultimately combine.

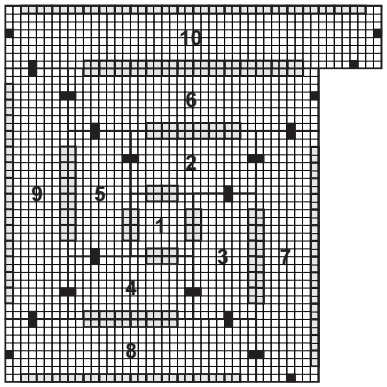

Figure 7 The first 10 rectangles weakly self-assembling an arbitrary computable pattern.

The darker grey portions represent the binary teeth used to connect the rectangles. Note that these rectangle-rectangle interfaces get larger as the rectangles grow out from the center, but since there remain unconnected portions of the perimeters of each rectangle, the final assembly is not fully connected. For infinite patterns, the portion of the construction that performs the Turing machine computation and outputs the definitions of the rectangles must be slightly modified so that the rectangles are formed on the left side of the north-growing simulation, enumerated one after another. This allows for an arbitrarily large portion of such a pattern to be weakly self-assembled by simply allowing the assembly to proceed for a "long enough" period of time before performing the *BREAK* operation.

References

1 Zachary Abel, Nadia Benbernou, Mirela Damian, Erik D. Demaine, Martin L. Demaine, Robin Flatland, Scott D. Kominers, and Robert T. Schweller, *Shape replication through self-assembly and rnase enzymes*, Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, 2010, pp. 1045–1064.

2 Leonard Adleman, *Toward a mathematical theory of self-assembly (extended abstract)*, Tech. Report 00-722, University of Southern California, 2000.

3 Leonard Adleman, Qi Cheng, Ashish Goel, and Ming-Deh Huang, *Running time and program size for self-assembled squares*, Proceedings of the thirty-third annual ACM Symposium on Theory of Computing, 2001, pp. 740–748.

4 Leonard Adleman, Qi Cheng, Ashish Goel, Ming-Deh Huang, and Hal Wasserman, *Linear self-assemblies: Equilibria, entropy and convergence rates*, In Sixth International Conference on Difference Equations and Applications, Taylor and Francis, 2001.

5 Robert D. Barish, Rebecca Schulman, Paul W. Rothemund, and Erik Winfree, *An information-bearing seed for nucleating algorithmic self-assembly*, Proceedings of the National Academy of Sciences **106** (2009), no. 15, 6054–6059.

6 Ho-Lin Chen and Ashish Goel, *Error free self-assembly with error prone tiles*, Proceedings of the 10th International Meeting on DNA Based Computers, 2004.

7 Ho-Lin Chen, Rebecca Schulman, Ashish Goel, and Erik Winfree, *Reducing facet nucleation during algorithmic self-assembly*, Nano Letters **7** (2007), no. 9, 2913–2919.

8 Qi Cheng, Gagan Aggarwal, Michael H. Goldwasser, Ming-Yang Kao, Robert T. Schweller, and Pablo Moisset de Espanés, *Complexities for generalized models of self-assembly*, SIAM Journal on Computing **34** (2005), 1493–1515.

9 Erik D. Demaine, Martin L. Demaine, Sándor P. Fekete, Mashhood Ishaque, Eynat Rafalin, Robert T. Schweller, and Diane L. Souvaine, *Staged self-assembly: nanomanufacture of arbitrary shapes with $O(1)$ glues*, Natural Computing **7** (2008), no. 3, 347–370.

10 Erik D. Demaine, Matthew J. Patitz, Robert T. Schweller, and Scott M. Summers, *Self-assembly of arbitrary shapes using rnase enzymes: Meeting the kolmogorov bound with small scale factor (extended abstract)*, Tech. Report 1004.4383, Computing Research Repository, 2010.

11 David Doty, *Randomized self-assembly for exact shapes*, SIAM Journal on Computing **39** (2010), no. 8, 3521–3552.

12 David Doty, Matthew J. Patitz, Dustin Reishus, Robert T. Schweller, and Scott M. Summers, *Strong fault-tolerance for self-assembly with fuzzy temperature*, Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science, 2010, pp. 417–426.

13 James I. Lathrop, Jack H. Lutz, Matthew J. Patitz, and Scott M. Summers, *Computability and complexity in self-assembly*, Theory of Computing Systems, to appear.

14 Furong Liu, Ruojie Sha, and Nadrian C. Seeman, *Modifying the surface features of two-dimensional DNA crystals.*, Journal of the American Chemical Society **121** (1999), no. 5, 917–922.

15 Chris Luhrs, *Polyomino-safe DNA self-assembly via block replacement*, Proceedings of The Fourteenth International Meeting on DNA Computing and Molecular Programming (DNA 14) (Ashish Goel, Friedrich C. Simmel, and Petr Sosík, eds.), Lecture Notes in Computer Science, vol. 5347, Springer, 2008, pp. 112–126.

16 Chengde Mao, Thomas H. LaBean, John H. Relf, and Nadrian C. Seeman, *Logical computation using algorithmic self-assembly of DNA triple-crossover molecules.*, Nature **407** (2000), no. 6803, 493–6.

17 Chengde Mao, Weiqiong Sun, , and Nadrian C. Seeman, *Designed two-dimensional DNA holliday junction arrays visualized by atomic force microscopy.*, Journal of the American Chemical Society **121** (1999), no. 23, 5437–5443.

18 Paul W. K. Rothemund and Erik Winfree, *The program-size complexity of self-assembled squares (extended abstract)*, Proceedings of the thirty-second annual ACM Symposium on Theory of Computing, 2000, pp. 459–468.

19 Paul W.K. Rothemund, Nick Papadakis, and Erik Winfree, *Algorithmic self-assembly of DNA Sierpinski triangles*, PLoS Biology **2** (2004), no. 12, 2041–2053.

20 David Soloveichik and Erik Winfree, *Complexity of self-assembled shapes*, SIAM Journal on Computing **36** (2007), no. 6, 1544–1569.

21 Paul Vitányi and Ming Li, *An introduction to kolmogorov complexity and its applications*, Springer, 1997.

22 Erik Winfree, *Algorithmic self-assembly of DNA*, Ph.D. thesis, California Institute of Technology, June 1998.

23 ———, *Simulations of computing by self-assembly*, Tech. Report CaltechCSTR:1998.22, California Institute of Technology, 1998.

24 ———, *Self-healing tile sets*, Nanotechnology: Science and Computation (Junghuei Chen, Natasa Jonoska, and Grzegorz Rozenberg, eds.), Natural Computing Series, Springer, 2006, pp. 55–78.

25 Erik Winfree and Renat Bekbolatov, *Proofreading tile sets: Error correction for algorithmic self-assembly.*, DNA (Junghuei Chen and John H. Reif, eds.), Lecture Notes in Computer Science, vol. 2943, Springer, 2003, pp. 126–144.

26 Erik Winfree, Furong Liu, Lisa A. Wenzler, and Nadrian C. Seeman, *Design and self-assembly of two-dimensional DNA crystals.*, Nature **394** (1998), no. 6693, 539–44.