Optimal Staged Self-Assembly of General Shapes*

Cameron Chalk[1], Eric Martinez[2], Robert Schweller[3], Luis Vega[4], Andrew Winslow[5], and Tim Wylie[6]

1 **Department of Computer Science, University of Texas – Rio Grande Valley, Brownsville, USA**
cameron.chalk01@utrgv.edu
2 **Department of Computer Science, University of Texas – Rio Grande Valley, Brownsville, USA**
eric.m.martinez02@utrgv.edu
3 **Department of Computer Science, University of Texas – Rio Grande Valley, Brownsville, USA**
robert.schweller@utrgv.edu
4 **Department of Computer Science, University of Texas – Rio Grande Valley, Brownsville, USA**
luis.a.vega01@utrgv.edu
5 **Département d'Informatique, Université Libre de Bruxelles, Brussels, Belgium**
awinslow@ulb.ac.be
6 **Department of Computer Science, University of Texas – Rio Grande Valley, Brownsville, USA**
timothy.wylie@utrgv.edu

Abstract

We analyze the number of stages, tiles, and bins needed to construct $n \times n$ squares and scaled shapes in the staged tile assembly model. In particular, we prove that there exists a staged system with b bins and t tile types assembling an $n \times n$ square using $\mathcal{O}(\frac{\log n - tb - t\log t}{b^2} + \frac{\log\log b}{\log t})$ stages and $\Omega(\frac{\log n - tb - t\log t}{b^2})$ are necessary for almost all n. For a shape S, we prove $\mathcal{O}(\frac{K(S) - tb - t\log t}{b^2} + \frac{\log\log b}{\log t})$ stages suffice and $\Omega(\frac{K(S) - tb - t\log t}{b^2})$ are necessary for the assembly of a scaled version of S, where $K(S)$ denotes the Kolmogorov complexity of S. Similarly tight bounds are also obtained when more powerful *flexible* glue functions are permitted. These are the first staged results that hold for all choices of b and t and generalize prior results. The upper bound constructions use a new technique for efficiently converting each both sources of system complexity, namely the tile types and mixing graph, into a "bit string" assembly.

1998 ACM Subject Classification F.1.1. Models of Computation

Keywords and phrases Tile self-assembly, 2HAM, aTAM, DNA computing, biocomputing

Digital Object Identifier 10.4230/LIPIcs.ESA.2016.26

1 Introduction

The *staged* self-assembly model is a generalization of the *two-handed* [1, 4, 7, 8] or *hierarchical* [5, 12] tile self-assembly models. In tile self-assembly, system monomers are unit squares with edge labels that collide randomly and attach permanently if abutting edge labels match sufficiently. This simple model is an abstraction of a DNA-based molecular implementation at

* Research supported in part by National Science Foundation Grants CCF-1117672, CCF-1555626, and CCF-1422152.

the nanoscale [13, 21] and is computationally universal [21]. The staged variant is motivated by experimental settings, where parallelism and mixing can be achieved (e.g. test tubes). Liquid-handling robots have been used to perform complex mixing instructions in the lab [15], similar to the mixing algorithms of staged self-assembly systems.

The staged model [8] extends the two-handed model by carrying out separate assembly processes in multiple *bins*. Assembly in each bin begins with *input assemblies* previously assembled in other bins. These bins are stratified into stages, and a *mix graph* specifies which bins in the previous stage supply each bin with input assemblies. The *output* of a staged self-assembly system is the set of assemblies produced in the bins of the final stage.

A common goal in the design of self-assembling systems is the construction of a desired shape. Here we consider the design of *efficient* systems with minimal complexity for a given shape. Three metrics exist for staged systems: the number of distinct tile types used in the system (*tile complexity*), the maximum number of bins used in any stage (*bin complexity*), and the number of stages (*stage complexity*). Efficient construction for various classes of shapes [8, 10] and patterns [9, 22] have been considered, and further extensions and variants of the staged self-assembly model have also been studied [1, 3, 11, 16, 17, 18].

Our results. Here we study the two classic benchmarks for the efficiency a tile self-assembly model: the assembly of $n \times n$ squares and arbitrary shapes (with scaling permitted). Previous works [19, 20, 2] achieved matching upper and lower (univariate) bounds on the minimum complexity of systems that assemble these shape classes in the very first tile assembly model [21]. Here we give nearly matching upper and lower (trivariate) bounds for assembling these shapes in the staged model; our results are summarized in Table 1.

For a given number of tile types t and bins b, we prove that any $n \times n$ square is constructed by a system with $\mathcal{O}(\frac{\log n - t \log t - tb}{b^2} + \frac{\log \log b}{\log t})$ stages and a scaled version[1] of any shape S is assembled by a system with $\mathcal{O}(\frac{K(S) - t \log t - tb}{b^2} + \frac{\log \log b}{\log t})$ stages, where $K(S)$ denotes the Kolmogorov complexity of S with respect to some fixed universal Turing machine. We pair these results with nearly matching lower bounds, proving that for almost all natural numbers n^2, $\Omega(\frac{\log n - t \log t - tb}{b^2})$ stages are needed to assemble an $n \times n$ square, and for all shapes S, $\Omega(\frac{K(S) - t \log t - tb}{b^2})$ stages are needed to assemble a scaled version of a given shape S.

We further explore the stage complexity of these shapes within the *flexible glue* model of tile attachment [6] (where non-matching glue labels can have strength), and prove that $n \times n$ squares and scaled shapes can be assembled using $\mathcal{O}(\frac{\log n - t^2 - tb}{b^2} + \frac{\log \log b}{\log t})$ and $\mathcal{O}(\frac{K(S) - t^2 - tb}{b^2} + \frac{\log \log b}{\log t})$ stages, respectively. We pair this with nearly matching lower bound stage complexities of $\Omega(\frac{\log n - t^2 - tb}{b^2})$ and $\Omega(\frac{K(S) - t^2 - tb}{b^2})$.

Our upper bounds both use a new technique to efficiently assemble *bit string pads*: constant-width assemblies with an exposed sequence of glues encoding a given bit string. This technique converts all three forms of system complexity (tile, bin, and stage) into bits of the string with only a constant-factor loss of information. In other words, the number of bits in the bit string pad rises linearly with the number of bits needed to specify the tile types and mix graph of the construction.

---

[1] The scale factor is proportional to the product of the time and space used by the fixed universal Turing machine to encode S using $K(S)$ bits.

[2] The fraction of values for which the statement holds reaches 1 in the limit as $n \to \infty$.

Table 1 The main results obtained in this work: upper and lower bounds on the number of stages of a staged self-assembly system with b bins and t tile types uniquely assembling $n \times n$ squares and scaled shapes. $K(S)$ denotes the Kolmogorov complexity of a shape.

Standard Glue Stage Complexity Results

Shape	Upper Bound	Theorem	Lower Bound	Theorem
$n \times n$	$\mathcal{O}(\frac{\log n - t \log t - tb}{b^2} + \frac{\log \log b}{\log t})$	11	$\Omega(\frac{\log n - t \log t - tb}{b^2})$	12
Scaled shapes	$\mathcal{O}(\frac{K(S) - t \log t - tb}{b^2} + \frac{\log \log b}{\log t})$	13	$\Omega(\frac{K(S) - t \log t - tb}{b^2})$	14

Flexible Glue Stage Complexity Results

$n \times n$	$\mathcal{O}(\frac{\log n - t^2 - tb}{b^2} + \frac{\log \log b}{\log t})$	20	$\Omega(\frac{\log n - t^2 - tb}{b^2})$	21
Scaled shapes	$\mathcal{O}(\frac{K(S) - t^2 - tb}{b^2} + \frac{\log \log b}{\log t})$	22	$\Omega(\frac{K(S) - t^2 - tb}{b^2})$	23

Comparison with prior work. In providing a class of nearly optimal staged systems for any choice of bin and tile count, our results also generalize and improve on prior results. For instance, Theorem 11 implies construction of $n \times n$ squares using $\mathcal{O}(1)$ bins, $\mathcal{O}(\frac{\log n}{\log \log n})$ tile types, and $\mathcal{O}(1)$ stages, matching a result of [2] (up to constant factors). For flexible glues, this is improved to $\mathcal{O}(\sqrt{\log n})$ tile types, a result of [6]. The same theorem also yields constructions using $\mathcal{O}(1)$ bins, $\mathcal{O}(1)$ tile types, and $\mathcal{O}(\log n)$ stages (matching a result of [8]) or $\mathcal{O}(\sqrt{\log n})$ bins, $\mathcal{O}(1)$ tile types, and $\mathcal{O}(\log \log \log n)$ stages, substantially improving over the $\mathcal{O}(\log \log n)$ stages used in [8]. For constructing scaled shapes, Theorem 13 implies systems using $\mathcal{O}(1)$ bins, $\mathcal{O}(\frac{K(S)}{\log K(S)})$ tile types, and $\mathcal{O}(1)$ stages, a result of [20].

2 The Staged Assembly Model

Tiles. A *tile* is a non-rotatable unit square with each edge labeled with a *glue* from a set Σ. Each pair of glues $g_1, g_2 \in \Sigma$ has a non-negative integer *strength*, denoted $\text{str}(g_1, g_2)$. Every set Σ contains a special *null glue* whose strength with every other glue is 0. If the glue strengths do not obey $\text{str}(g_1, g_2) = 0$ for all $g_1 \neq g_2$, then the glues are *flexible*. Unless otherwise stated, we assume that glues are not flexible.

Configurations, assemblies, and shapes. A *configuration* is a partial function $A : \mathbb{Z}^2 \to T$ for some set of tiles T, i.e., an arrangement of tiles on a square grid. For a configuration A and vector $\vec{u} = \langle u_x, u_y \rangle \in \mathbb{Z}^2$, $A + \vec{u}$ denotes the configuration $f \circ A$, where $f(x, y) = (x + u_x, y + u_y)$. For two configurations A and B, B is a *translation* of A, written $B \simeq A$, provided that $B = A + \vec{u}$ for some vector $\vec{u}$. For a configuration A, the *assembly* of A is the set $\tilde{A} = \{B : B \simeq A\}$. The *shape* of an assembly $\tilde{A}$ is $\{\text{dom}(A) : A \in \tilde{A}\}$ where dom() is the domain of a configuration. A shape S' is a *scaled* version of shape S provided that for some $k \in \mathbb{N}$ and $D \in S$, $\bigcup_{(x,y) \in D} \bigcup_{(i,j) \in \{0,1,\ldots,k-1\}^2} (kx + i, ky + j) \in S'$.

Bond graphs and stability. For a configuration A, define the *bond graph* G_A to be the weighted grid graph in which each element of $\text{dom}(A)$ is a vertex, and the weight of the edge between a pair of tiles is equal to the strength of the coincident glue pair. A configuration is τ-*stable* for $\tau \in \mathbb{N}$ if every edge cut of G_A has strength at least τ, and is τ-*unstable* otherwise. Similarly, an assembly is τ-*stable* provided the configurations it contains are τ-stable. Assemblies $\tilde{A}$ and $\tilde{B}$ are τ-*combinable* into an assembly $\tilde{C}$ provided there exist $A \in \tilde{A}$, $B \in \tilde{B}$, and $C \in \tilde{C}$ such that $A \cup B = C$ and $\tilde{C}$ is τ-stable.

Two-handed assembly and bins. We define the assembly process via bins. A bin is an ordered tuple (S, τ) where S is a set of *initial* assemblies and $\tau \in \mathbb{N}$ is the *temperature*. In this work, τ is always equal to 2. For a bin (S, τ), the set of *produced* assemblies $P'_{(S,\tau)}$ is defined recursively as follows:

1. $S \subseteq P'_{(S,\tau)}$.
2. If $A, B \in P'_{(S,\tau)}$ are τ-combinable into C, then $C \subseteq P'_{(S,\tau)}$.

A produced assembly is *terminal* provided it is not τ-combinable with any other producible assembly, and the set of all terminal assemblies of a bin (S, τ) is denoted $P_{(S,\tau)}$. That is, $P'_{(S,\tau)}$ represents the set of all possible supertiles that can assemble from the initial set S, whereas $P_{(S,\tau)}$ represents only the set of supertiles that cannot grow any further.

If all assemblies in $P'_{(S,\tau)}$ have finite size, then the assemblies in $P_{(S,\tau)}$ are *uniquely* produced by bin (S, τ). Unique production implies that every producible assembly can be repeatedly combined with others to form an assembly in $P_{(S,\tau)}$.

Staged assembly systems. An *r-stage b-bin mix graph* M is an acyclic r-partite digraph consisting of rb vertices $m_{i,j}$ for $1 \leq i \leq r$ and $1 \leq j \leq b$, and edges of the form $(m_{i,j}, m_{i+1,j'})$ for some i, j, j'. A *staged assembly system* is a 3-tuple $\langle M_{r,b}, \{T_1, T_2, \ldots, T_b\}, \tau \rangle$ where $M_{r,b}$ is an r-stage b-bin mix graph, T_i is a set of tile types, and $\tau \in \mathbb{N}$ is the temperature. Given a staged assembly system, for each $1 \leq i \leq r$, $1 \leq j \leq b$, a corresponding bin $(R_{i,j}, \tau)$ is defined as follows:

1. $R_{1,j} = T_j$ (this is a bin in the first stage);
2. For $i \geq 2$, $R_{i,j} = \Big(\bigcup_{k:\ (m_{i-1,k}, m_{i,j}) \in M_{r,b}} P_{(R_{(i-1,k)}, \tau_{i-1,k})} \Big)$.

Thus, bins in stage 1 are tile sets T_j, and each bin in any subsequent stage receives an initial set of assemblies consisting of the terminally produced assemblies from a subset of the bins in the previous stage as dictated by the edges of the mix graph.[3] The *output* of a staged system is the union of the set of terminal assemblies of the bins in the final stage.[4] The output of a staged system is *uniquely produced* provided each bin in the staged system uniquely produces its terminal assemblies.

3 Key Lemmas

Our results rely on two key lemmas. The first is an upper bound on the information content of a staged system that implies the lower bounds on system complexity. The second is a formal statement of the previously mentioned bit string pad construction.

▶ **Lemma 1.** *A staged system of fixed temperature τ with b bins, s stages, and t tile types can be specified using $\mathcal{O}(t \log t + sb^2 + tb)$ bits. Such a system with flexible glues can be specified using $\mathcal{O}(t^2 + sb^2 + tb)$ bits.*

[3] The original staged model [8] only considered $\mathcal{O}(1)$ distinct tile types, and thus for simplicity allowed tiles to be added at any stage (since $\mathcal{O}(1)$ extra bins could hold the individual tile types to mix at any stage). Because systems here may have super-constant tile complexity, we restrict tiles to only be added at the initial stage.

[4] This is a slight modification of the original staged model [8] in that there is no requirement of a final stage with a single output bin. It may be easier in general to solve problems in this variant of the model, so we consider it for lower bound purposes. However, all of our results apply to both variants of the model.

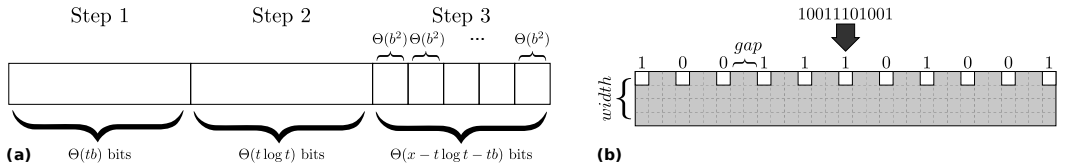

Figure 1 (a) The decomposition of a bit string pad's bits into those encoded by the three steps of a staged system with t tile types and b bins. (b) An example bit string $r = 10011101001$ encoded as a *width*-4 *gap*-2 11-*bit string pad* where the top glues correspond to the bits in r.

Proof. A staged system can be specified in four parts: the tile types, the glue function, the mix graph, and the assignment of tile types to stage-1 bins. We separately bound the number of bits required to specify each.

A set of t tile types has up to $4t$ glue types, so specifying each tile requires $\mathcal{O}(\log t)$ bits, and the entire tile set takes $\mathcal{O}(t \log t)$ bits. If the system does not have flexible glues, then the glue function can be specified in $\mathcal{O}(4t) = \mathcal{O}(t)$ bits, using $\mathcal{O}(\log \tau) = \mathcal{O}(1)$ bits per glue type to specify the glue's strength. If the system has flexible glues, then the glue function can be specified using $\mathcal{O}(1)$ bits per pairwise glue interaction and $\mathcal{O}((4t)^2) = \mathcal{O}(t^2)$ bits total.

The mix graph consists of bs nodes. Each pair of nodes in adjacent stages optionally share a directed edge pointing upwards. Thus specifying these edges takes $\mathcal{O}(b^2(s-1)) = \mathcal{O}(b^2 s)$ bits. The assignment of tile types to stage-1 bins requires one bit per each choice of tile type and bin, or $\mathcal{O}(tb)$ bits total.

Thus a staged system without flexible glues can be specified in $\mathcal{O}(t \log t + t + b^2 s + tb)$ bits, and otherwise in $\mathcal{O}(t \log t + t^2 + b^2 s + tb)$ bits. ◀

It immediately follows from Lemma 1 that for most bit strings of length x, any staged system with b bins and t tiles that encodes the bit string must have $\Omega(\frac{x - tb - t \log t}{b^2})$ stages with standard glues and $\Omega(\frac{x - tb - t^2}{b^2})$ stages with flexible glues.

The two main positive results of this work, efficient assembly of squares and general scaled shapes, both rely mainly on efficient assembly of *bit string pads*: assemblies that expose a sequence of north glues that encode a bit string. An example is shown in Figure 1(b). Squares and general scaled shapes are assembled by combining a universal set of "computation" tiles with efficiently assembled "input" bit string pads.

▶ **Definition 2** (bit string pad). A *width-k gap-f r-bit string pad* is a $k \times (f(r-1)+1)$ rectangular assembly with r glues from a set of two glue types $\{0, 1\}$ exposed on the north face of the rectangle at intervals of length f, starting from the leftmost north edge. Unless otherwise specified, a bit string pad is gap-0. All remaining exposed glues on the north tile edges have some common label f. The remaining exposed south, east, and west tile edges have glues g_S, g_E, and g_W. A bit string pad *represents* a given string of r bits if the exposed "0" and "1" glues from left to right are equal to the given bit string.

Bit string pads are constructed by decomposing the pad into three subpads and constructing each in a separate step using a different source of system complexity (see Figure 1(a)):

- Step 1: $\Theta(tb)$ bits from assigning tile types to stage-1 bins (Section 4.2).
- Step 2: $\Theta(t \log t)$ bits from the tile types themselves as in [2, 6, 14, 20] (Section 4.3).
- Step 3: $\Theta(x - t \log t - tb)$ bits from the mix graph using a variant of "crazy mixing" [8] (Section 4.4).

These subpads are then combined into the complete pad. If flexible glues are permitted, Step 2 is modified as in [6] to achieve $\mathcal{O}(\frac{x - tb - t \log t}{b^2} + \frac{\log \log b}{\log t})$ stages.

▶ **Lemma 3.** *There exist constants $c, d \in \mathbb{N}$ such that, for any $t, b \in \mathbb{N}$ with $t > c$, $b > d$ and bit string S of length x, there exists a staged system with b bins, t tiles, and $\mathcal{O}(\frac{x-tb-t\log t}{b^2} + \frac{\log\log b}{\log t})$ stages that assembles a width-9 gap-$\Theta(\log b)$ l-bit string pad representing S.*

Proof. Let $t' = \frac{t-10}{4}$ and $b' = \frac{b-10}{4}$. Using an approach similar to that in Section 4.1, construct a length $1 \times 2\log\frac{b'-15}{9} + 2$ *filler assembly* using t' tile types and b' bins in $\mathcal{O}(\frac{\log\log b'}{\log t'})$ stages such that the assembly has glue e on its west edge (matching that of the east side of the bit string pads) and glue w on its east edge. Next, use Lemma 5 with t' tile types, b' bins, and $\mathcal{O}(\frac{\log\log b'}{\log t'})$ stages to construct a width-9 gap-$2\log\frac{b'-15}{9} + 2$ $\Theta(tb)$-bit string pad. Then use Lemma 9 with t' tile types, b' bins, and $\mathcal{O}(1)$ stages to construct a width-3, gap-$2\log\frac{b'-15}{9} + 2$, $\Theta(t\log t)$-bit string pad.

So far, $\Theta(tb) + \Theta(t\log t)$ bits have been encoded and so $\Theta(x - tb - t\log t)$ bits remain. Invoke Lemma 10 with t' tile types, b' bins, and $\mathcal{O}(\frac{x-tb-t\log t}{b^2} + \frac{\log\log b'}{\log t'})$ stages to construct a width-9 gap-$2\log\frac{b'-15}{9} + 2$ $\Theta(x - tb - t\log t)$-bit string pad. In one final stage, concatenate two bit string pads using the filler assembly and in one more stage concatenate the third.

By concatenating the length $\Theta(tb)$-bit string pad, the length $\Theta(t\log t)$-bit string pad, and the $\Theta(x - tb - t\log t)$-bit string pad, each separated by the $2\log\frac{b'-15}{9} + 2$ filler assembly, an x-bit string pad with $\mathcal{O}(\log b)$ spacing is constructed; use 10 additional tile types (in 10 bins) to "fill in" the portions of the assembly with width less than 9.

The total number of tile types and bins used are $4t' + 10 = t$ and $4b' + 10 = b$, respectively, with $4t'$ and $4b'$ used for the three bit string pads and one connector assembly and the remainder for filling in the pad to width 9. The total number of stages used is $\mathcal{O}(\frac{\log\log b'}{\log t'}) + \mathcal{O}(1) + \mathcal{O}(\frac{x-tb-t\log t}{b^2} + \frac{\log\log b'}{\log t'}) = \mathcal{O}(\frac{\log b}{t} + \frac{x-tb-t\log t}{b^2})$. ◀

The additive gap between the upper and lower bounds implied by these lemmas comes from the $\mathcal{O}(\frac{\log\log b}{\log t})$ additional stages used to construct some of the machinery needed to carry out the three steps of Lemma 3.

4 Bit String Pad Construction

As mentioned, bit string pads are assembled by combining three subpads constructed via separate and independent methods that utilize distinct sources of information complexity in a staged self-assembly system. Each subpad encodes a number of bits roughly proportional to the number needed to describe the corresponding portion of the staged system, i.e., an asympotitically optimal number of bits are encoded.

4.1 Wings

The additive gap in our upper and lower bounds come from a helpful subconstruction used in Steps 1 and 3 described here. This subconstruction assembles all 1-gapped width-2 bit string pads of a given length in separate bins:

▶ **Lemma 4.** *There exist constants $c, d \in \mathbb{N}$ such that, for any $t, b \in \mathbb{N}$ with $t > c$ and $b > d$, there is a staged self-assembly system with b bins, t tile types, and $\mathcal{O}(\frac{\log\log b}{\log t})$ stages that assembles all gap-1, width-2, $\log(b)$-bit string pads, each placed in a distinct bin.*

Due to space constraints, the proof of this and some later results are omitted. We give proof sketches instead. Let $\gamma = \lfloor\frac{t-6}{2}\rfloor$ and $\eta = \gamma + 1$. If $\gamma \geq 2\log(b)$, directly build all the bit string pads in $\mathcal{O}(1)$ stages. Otherwise, repeatedly apply a constant-stage "round" that

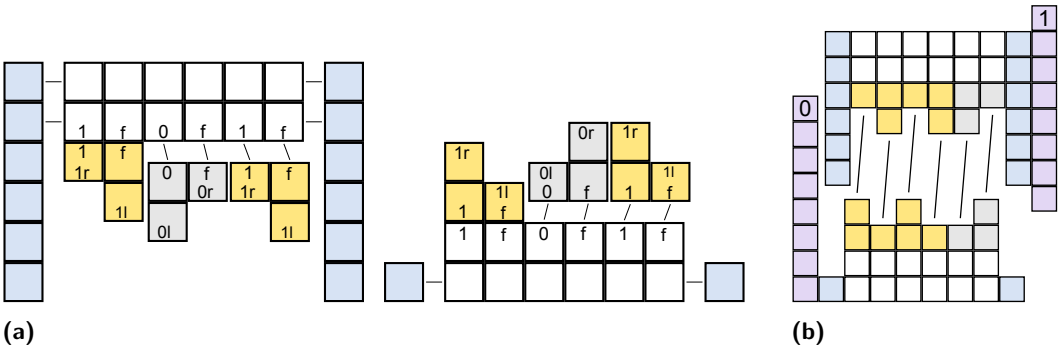

Figure 2 **(a)** The attachment of extra subassemblies onto bit string pads to create left and right wings. Each of the two size 3 subassemblies use 3 tiles to deterministically assemble the respective L shape in their own bins. **(b)** The attachment of two bit strips using matching wings. Note that the geometry attached to the sides of each wing prevent misaligned, non-matching wings to attach.

starts with all binary gadgets of a given length and yields all binary gadgets of a factor of η longer, starting with just two bit string pads encoding the two bit strings of length 1.

Use $\mathcal{O}(1)$ additional tile types, bins, and stages to augment the the bit string pads assembled by Lemma 4 into left and right *wings* (seen in the left and right portions of Figure 2(a)) that attach when the underlying bit strings are identical. These wings are used in Steps 1 and 3 to achieve ordered assembly of bit string subpads into larger bit string pads.

4.2 Step 1: encoding via initial tile-to-bin assignment

Recall that in a staged system, each of the system's b stage-1 bins is assigned a subset of t total tile types. Here we design an assignment that assembles a $\Theta(tb)$-bit string subpad of the final bit string pad using $\mathcal{O}(\log\log b/\log t)$ stages - enough to utilize the wings of Section 4.1. The assignment yields b bins that contain assemblies encoding distinct equal-length substrings of the $\Theta(tb)$ bits. These assemblies are then combined using wings.

▶ **Lemma 5.** *There exist $c, d \in \mathbb{N}$ such that, for all $t, b \in \mathbb{N}$ with $t > c$ and $b > d$ and bit string S of length $\Theta(tb)$, there is a staged self-assembly system with b bins, t tiles, and $\mathcal{O}(\frac{\log\log b}{\log t})$ stages that assembles a gap-$(2\log\lfloor\frac{b-15}{9}\rfloor + 2)$ $\Theta(tb)$-bit string pad representing S.*

See Figure 3 for a sketch of the idea. Let γ and β be constant fractions of t and b, respectively. Use γ tiles and β bins to construct all left and right $\log(\beta)$-bit wings according to Section 4.1. Also construct $\frac{\gamma}{2}$ constant-sized *bit strip* subassemblies that expose a 0 or 1 north glue and have wings attached to their right and left sides such that any $\frac{\gamma}{2}$-bit string pad can be assembled from $\frac{\gamma}{2}$ bit strips attached sequentially.

In each of β bins, assemble $\frac{\gamma}{2}$ bit strips into a distinct $\frac{\gamma}{2}$-bit string subpad of the desired pad. Combine these β subpads with wings that encode their locations in the pad, and then combine these "wing-labeled" subpads to assemble the complete $\Theta(tb)$-bit string pad. The number of stages used is $\mathcal{O}(\frac{\log\log b}{\log t})$ (for the wings, see Lemma 4) plus $\mathcal{O}(1)$ (the subpads of the desired pad).

4.3 Step 2: encoding via tile types

Here the goal is to design a collection of t tile types that encodes $\Theta(t\log t)$ bits. The solution is to utilize the base conversion approach of [2, 6, 14, 20]. In this approach, tile

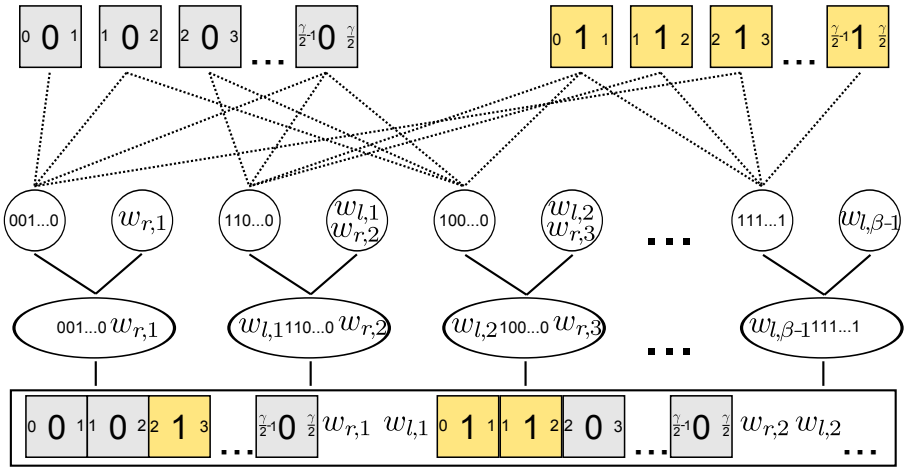

Figure 3 The creation of $\gamma\beta$-bit string pads. The squares labeled 0 and 1 represent bit strips. The dotted lines indicate tile to bin assignments before the first stage of the system; $w_{r,i}$ and $w_{l,i}$ represent the i^{th} right and left wings respectively.

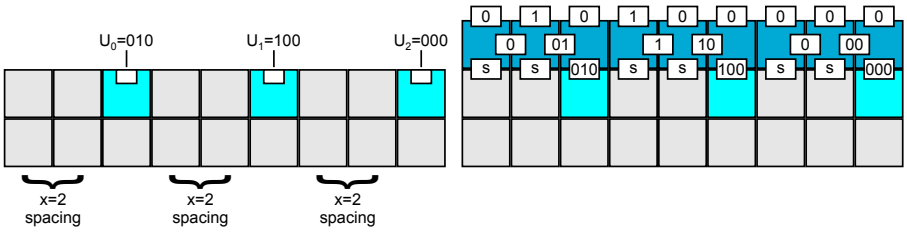

Figure 4 Left: a width-2 gap-$\log z - 1$ decompression pad representing a bit string $S = 010100000$ in base $z = 8$. Right: $\mathcal{O}(z)$ decompression tiles interact with the north glues of the decompression pad to combine into a width-3 bit string pad representing S in base 2.

types optimally encode integer values in a high base and then "decompressed" into a binary representation. In total, t tile types are used to encode (in a high base) and decompress (into a binary) $\Theta(t \log t)$ bits.

▶ **Definition 6** (decompression pad)**.** For $k, r, x \in \mathbb{N}$ and $u = 2^x$, a width-k, r-digit, base-u decompression pad is a $k \times rx$ rectangular assembly with r glues from a set of $u - 1$ glue types $\{0, 1, ..., u - 1\}$ exposed on the north face of the rectangle at intervals of length $x - 1$ and starting from the leftmost northern edge. All remaining glues on the north surface have a common type n. The remaining exposed south, east, and west tile edges have glues g_S, g_E, and g_W. A decompression pad *represents* a given string of digits in base u if the exposed glues from left to right, disregarding glues of type n, are equal to the given digit string in base u.

Consider the following example, also seen in Figure 4). Let $S = 010100000$ ($S = 240$ in base 8) be a bit string, with the goal of constructing a width-3 9-bit string pad representing S. First, build a decompression pad representing S in base 8 by combining 3 different $3 \times \log_2(8)$ blocks. Then convert the decompression pad into a bit string pad representing S using $\mathcal{O}(z)$ tile types.

▶ **Lemma 7.** *Given integers $x \geq 3$, $d \geq 1$ and $z = 2^x$, there exists a 1-stage, 1-bin staged self-assembly system that assembles a d-digit decompression pad of width-2 and base-z, using at most $5d + \log z - 2$ tile types.*

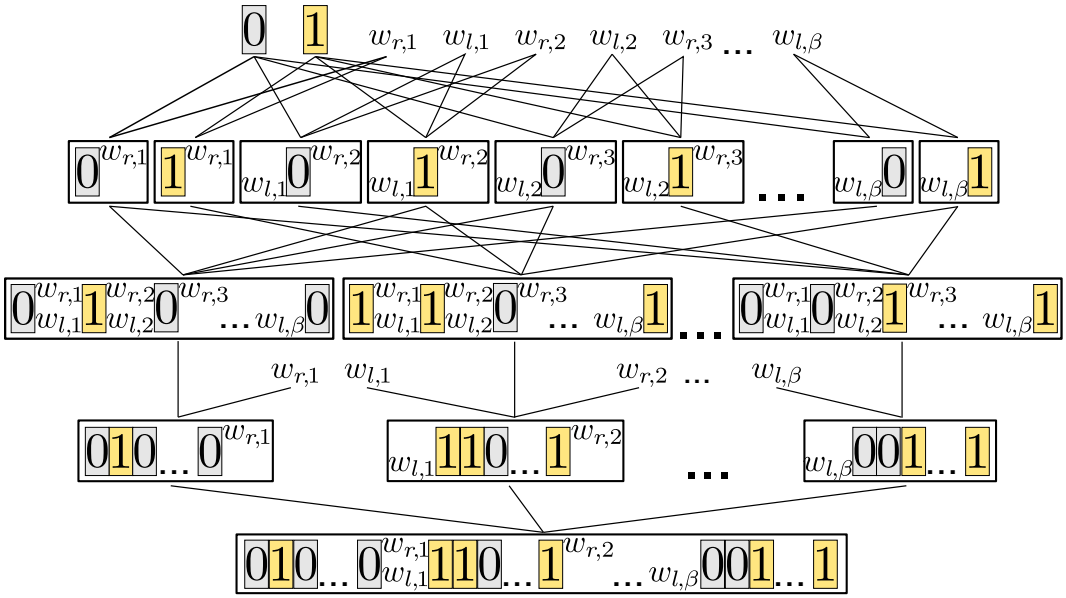

Figure 5 The creation of β^2-bit string pads using β wings and $\mathcal{O}(1)$ stages. The rectangles 0 and 1 represent bit strips that may attach wings on either side; $w_{r,i}$ and $w_{l,i}$ represent the i^{th} right and left wings respectively.

▶ **Lemma 8.** *Given integers $d \geq 3$, $x \geq 3$, $z = 2^x$, and bit string S of length $d\log(z)$, there exists a staged self-assembly system with 1 bin, $5d + 2z + \log z - 4$ tile types, and 1 stage that assembles a width-3 $d\log(z)$-bit string pad representing S.*

▶ **Lemma 9.** *There exists some constant $c \in \mathbb{N}$ such that, for any $t \geq c$ and bit string S of length $\Theta(t \log t)$, there exists a staged self-assembly system with 1 bin, t tile types, and 1 stage assembling a width-3 $\Theta(t \log t)$-bit string pad representing S.*

Omitted additional details are needed to convert these gap-0 pads to higher-gap pads consistent with those assembled in Section 4.4.

4.4 Step 3: encoding via mix graph

This step uses a mix graph to encode encodes a achieves the following efficient assembly:

▶ **Lemma 10.** *There exist $c, d \in \mathbb{N}$ such that, for any $t > c$ and $b > d$ and bit string S of length x, there is a staged self-assembly system with t bins, b tile types, and $\mathcal{O}(\frac{x}{b^2} + \frac{\log\log b}{\log t})$ stages that assembles a width-9 gap-$(2\log\frac{b-15}{9} + 2)$ x-bit string pad representing S.*

An overview of the construction is shown in Figure 5. Let γ and β represent some constant fractions of t and b respectively. Utilize γ tiles and β bins to construct all length-$\log_2(\beta)$ left and right wings according to Section 4.1 and denote the i^{th} left and wings by $w_{l,i}$ and $w_{r,i}$, respectively. Also construct two constant-sized *bit strip* subassemblies that expose a 0 or 1 north glue and allow wings to be attached to their right and left sides.

In the first stage and for all $1 \leq i \leq \beta$, mix $w_{r,i}$ and $w_{l,i-1}$ with bit strip 0 into a bin denoted b_i^0. Similarly, mix $w_{r,i}$ and $w_{l,i-1}$ with bit strip 1 into a bin denoted b_i^1 for a total of 2β bins.

In the second stage, selectively mix specific 0 or 1 winged bit strips to assemble specific β-bit string pads across β bins. Specifically, mix either b_i^0 or b_i^1 for each i across β bins for a total of β different β-bit string pads.

In the third stage, attach wings to each of the β-bit string pads. For each of the β bins, mix $w_{r,i}$ and $w_{l,i-1}$ into the bins such that $w_{r,1}$ is mixed with the first β bits of the desired β^2-bit string pad, $w_{r,2}$ and $w_{l,1}$ are mixed with the second β bits of the desired β^2-bit string pad, etc.

In the final stage, mix all β bins, each containing a β-bit string pads, into a common bin to create β^2-bit string pads. The wings ensure that the bit string pads attach in the desired order. Repeat this process $\frac{x}{\beta^2}$ times, each time concatenating the β^2-bit string pad onto each preceding bit string pad. In the end, a single x-bit string pad results. In total, $\mathcal{O}(\frac{\log\log b}{\log t})$ stages are used to construct the wings and $\mathcal{O}(\frac{x}{b^2})$ stages are used to assemble $\frac{x}{\beta^2}$ unique β^2-bit string pads. Thus this step has total stage complexity of $\mathcal{O}(\frac{x}{\beta^2} + \frac{\log\log\beta}{\log t}) = \mathcal{O}(\frac{x}{b^2} + \frac{\log\log b}{\log t})$.

5 Assembly of $n \times n$ Squares

Efficient assembly of $n \times n$ squares is obtained by combining bit string pads with a technique of Rothemund and Winfree [19]. Their technique utilizes a binary counting mechanism which constructs a length $\Theta(n)$ rectangle with $\Theta(\log n)$ width. The mechanism uses $\mathcal{O}(\log n)$ tile types to seed the counter at a certain value, and then $\mathcal{O}(1)$ tile types attach in a "zig-zag" pattern, where "zigs" copy the value from the row below and "zags" increment the the value by 1. Once the binary counter increments to its maximum value (a string of 1), the assembly stops growing. Two rectangles assembled this way can be combined to form a bounding box that is then filled to form a square. We utilize the bit string pad construction of Section 4 to efficiently assemble the seed for the binary counting mechanism, requiring only an additional $\mathcal{O}(1)$ tile types and 1 stage to perform the binary counting and square filling.

▶ **Theorem 11.** *There exist constants $b_0, t_0 \in \mathbb{N}$ such that for any $b, t, n \in \mathbb{N}$ with $b \geq b_0$, $t \geq t_0$, there exists a staged self-assembly system with b bins, t tile types, and $\mathcal{O}(\frac{\log n - t\log t - tb}{b^2} + \frac{\log\log b}{\log t})$ stages that uniquely produces an $n \times n$ square.*

Proof. Let c be the (constant) number of tile types used to implement the fixed-width "zig-zag" binary counting mechanism shown in [19]. Let $t' = t - c$, $b' = b - 2$, and $n' = \lceil \log n \rceil$. Let $m = 2^{n'-1} - (n-22)/2 - n'(2\log b' + 2)$. Using Lemma 3, construct two $\Theta(\log b)$-gap $\lceil \log m \rceil$-bit string pads encoding m, where each construction each uses b' bins, t' tile types and $\mathcal{O}(\frac{\lceil \log m \rceil - tb - t\log t}{b^2} + \frac{\log\log b}{\log t})$ stages. Figure 6 shows the construction, including modifications to the technique shown in [19].

On both pads, a small modification is made: the glues of the first and last bits are made unique and the first bit's glue strength is set to 2. This modification is necessary to implement a fixed-width binary counting mechanism as in [19] and uses $\mathcal{O}(1)$ additional tile types. Also, on the north-facing (east-facing) bit string pad, a unique strength-2 glue C2 is placed on the south (west) face of the pad's bottommost rightmost (topmost leftmost) tile. This special glue is used to combine the two pads with a unique tile type.

Note that the bit string pads assembled in Section 4 have substantial spacing between the exposed binary glues, but the counter of [19] has spacing 0. This is resolved by adding generic tiles which transfer information horizontally. These generic tiles use cooperative binding between a south-facing f glue (which matches the glue that spaces the bits on the bit string pad) and west/east glues representing the information to be passed horizontally across spacing of f glues. The tiles also expose a north-facing f glue to be used when the

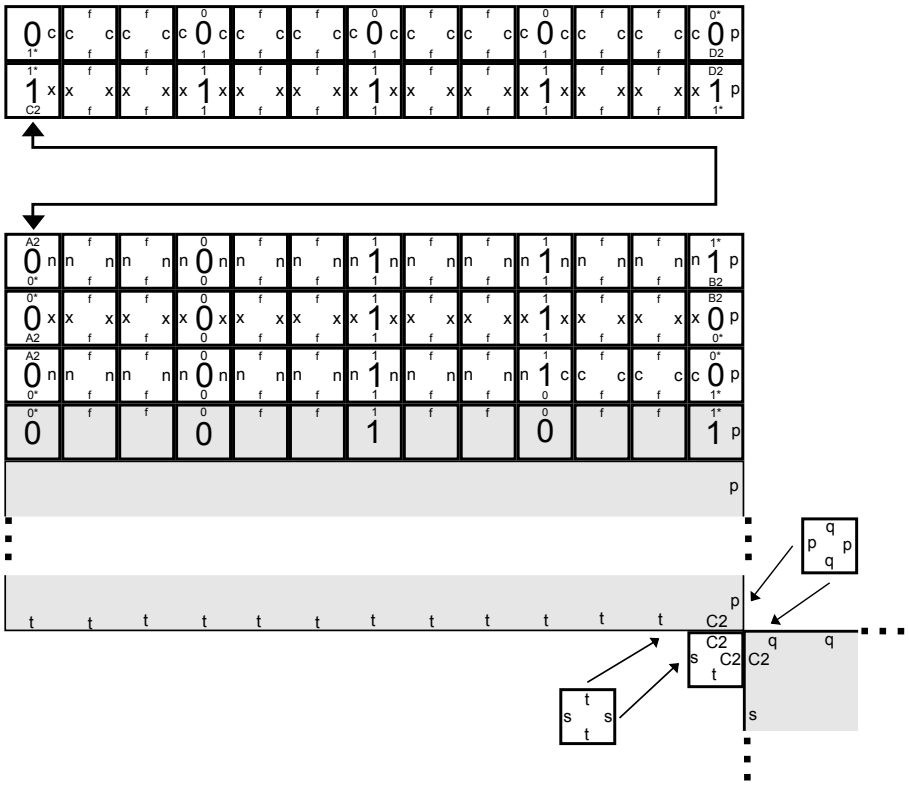

Figure 6 Constructing a counter seed. The bit string pads are shown in gray. Glues with a "2" in the string have strength-2, all other glues have strength 1.

information needs to be transferred across the spacing in the row above. Without loss of generality, rotated versions of these tiles are used in the east-growing counter.

The stage complexity of the system is $\mathcal{O}(\frac{\lceil \log n \rceil - tb - t \log t}{b^2} + \frac{\log \log b}{\log t})$. Note that the length of the bit string pads assembled according to Lemma 3 is dependent on b, the number of bins used to construct the bit string pad. If b is so large that the spacing between bits causes the width of the bit string pad to exceed n (roughly $\log b > n$), we instead directly construct the appropriate bit string pad with spacing 0 using $\mathcal{O}(\frac{\log \log b}{\log t})$ stages. ◀

The following lower bound is derived from Lemma 1 by observing that for almost all $n \in \mathbb{N}$, $\lfloor \log n \rfloor$ bits are needed to represent n.

▶ **Theorem 12.** *For any $b, t \in \mathbb{N}$ and almost all $n \in \mathbb{N}$, any staged self-assembly system which uses at most b bins and t tile types that uniquely assembles an $n \times n$ square must use $\Omega(\frac{\log n - t \log t - tb}{b^2})$ stages.*

6 Assembly of Scaled Shapes

Efficient assembly of arbitrary shapes (up to scaling) is achieved by combining bit string pads with the shape-building scheme of Soloveichik and Winfree [20]. Their construction uses two subsets of tile types: a varying set to encode the binary description of the target shape and a fixed set to decode the binary description and build the shape. We replace the first set with a bit string pad encoding the same information.

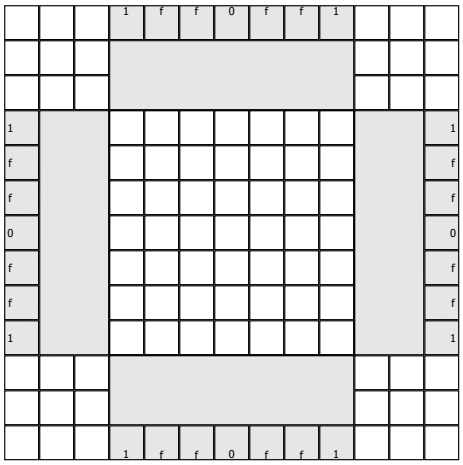

Figure 7 Construction of the modified seed block. Bit string pads are colored in gray. We concatenate four $K(S)$-bit string pads representing S.

▶ **Theorem 13.** *There exist constants $b_0, t_0 \in \mathbb{N}$ such that for any shape S of Kolmogorov complexity $K(S)$ and $b, t \in \mathbb{N}$ such that $b \geq b_0$ and $t \geq t_0$, there exists a staged self-assembly system with b bins, t tile types, and $\mathcal{O}(\frac{K(S)-t\log t-tb}{b^2} + \frac{\log\log b}{\log t})$ stages that uniquely produces S at some scale factor.*

Proof. Observe that the tile set described in [20] uniquely constructs the same terminal assembly, namely a *scaled* version of S where each cell is replaced by a square block of cells, when run at temperature 2 in the two-handed mixing model. It does so via a Kolmogorov-complexity-optimal Turing machine simulation of a machine that computes a spanning tree of the shape given a seed assembly or *seed block* encoding the shape. The simulation is then run as it "fills in" the shape, beginning with the seed block. Here a similar seed block is constructed and consists of four bit string pads, a square "core" and additional filler tiles.

Let $t' = \frac{t-c}{5}$ where c is the (constant) number of tile types required by [20] to carry out the simulation of a (fixed) universal Turing machine. Let $b' = \frac{b-1}{5}$.

Use the method of Lemma 3 to construct the modified seed block by assembling four different $K(S)$-bit string pads representing a program that outputs S, each using b' bins, t' tile types and $\mathcal{O}(\frac{K(S)-t'\log t'-t'b'}{b'^2} + \frac{\log\log b'}{\log t'})$ stages. These four pads (each with dimensions $(2K(S)\log K(S)+2) \times \mathcal{O}(1)$) are attached to the four sides of a $(2K(S)\log K(S)+2) \times (2K(S)\log K(S)+2)$ square constructed as in Theorem 11 using t' tile and b' bins in $\mathcal{O}(\frac{\log(2K(S)\log K(S)+2)-t'\log t'-t'b'}{b'^2} + \frac{\log\log b'}{\log t'})$ stages. An abstract figure of the completed seed block can be seen in Figure 7. The Turing simulation occurs in one stage by mixing the four concatenated bit string pads into one bin which contains the fixed set of Turing-machine-simulation tiles of [20]. The bit string pads contain spacing between the exposed binary glues, while the simulation tile types of [20] expect adjacent glues. This is resolved by modifying the Turing-machine-simulation tile set to include generic tiles for transferring information across spacing, similar to the tiles of the same purpose discussed in the proof of Theorem 11. We need at most 1 such tile for each tile in the (constant-sized) Turing-machine-simulation tile set, for a constant increase in tile complexity. The stage complexity is $4 \times \mathcal{O}(\frac{K(S)-t'\log t'-t'b'}{b'^2} + \frac{\log\log b'}{\log t'}) + \mathcal{O}(\frac{\log(2K(S)\log K(S)+2)-t'\log t'-t'b'}{b'^2} + \frac{\log\log b'}{\log t'}) = \mathcal{O}(\frac{K(S)-t\log t-tb}{b^2} + \frac{\log\log b}{\log t})$. ◀

The following theorem follows from the information-theoretic bound of Lemma 1.

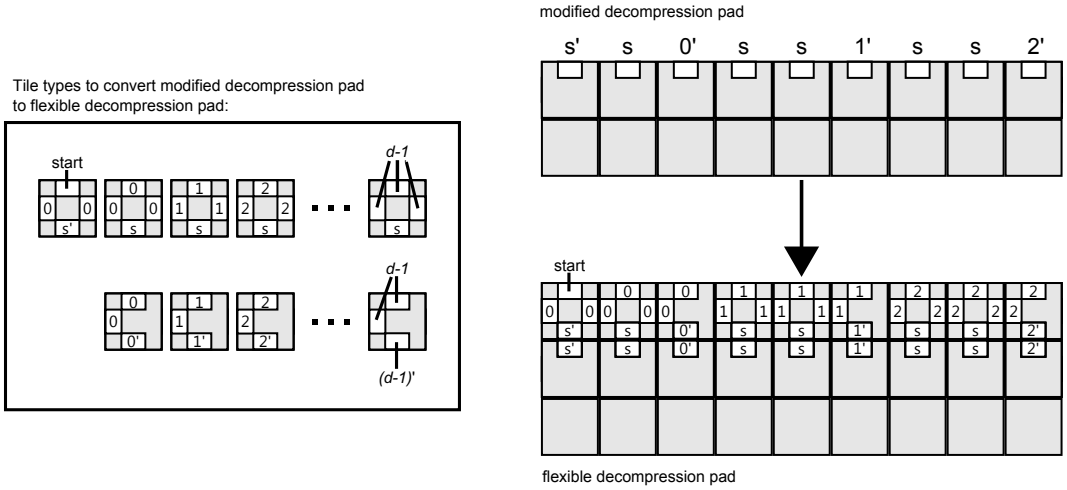

Figure 8 The templates to convert a modified decompression pad to a flexible decompression pad using $2d+1$ tile types, where integer $d \geq 3$, on the left. Using these additional tile types, a modified decompression pad is converted into a flexible decompression pad. A *modified decompression pad* has a westmost northmost glue of s' and every non-s glue on the north surface is a special *prime* version distinct from other similar glue types. On the left, a width-2 modified decompression pad representing the string 012 in base-8 is converted to a width-3 length-9 flexible decompression pad.

▶ **Theorem 14.** *For any $b, t \in \mathbb{N}$ and shape S with Kolmogorov complexity $K(S)$, any staged self-assembly system which uses at most b bins and t tile types that uniquely assembles S must use $\Omega(\frac{K(S)-t\log t-tb}{b^2})$ stages.*

7 Flexible Glues

Here, an alternate model permitting non-diagonal glue functions, also called *flexible glues* is considered. By modifying Step 2 of the bit string pad construction of Section 4 to encode $\Theta(t^2)$ bits rather than $\Theta(t\log(t))$ bits in t tile types, similarly tight results are obtained for the same problems in this more powerful model. The technique uses a modified decompression bad, similar to the technique introduced in [6].

▶ **Definition 15** (flexible decompression pad)**.** A width-k length-r^2 flexible decompression pad is a $k \times r^2$ rectangular assembly with r^2 north glue types from the set $\{\text{start}, 0, 1, \ldots, r\}$ exposed on the north face of the rectangle. The westmost glue is "start", the following $r-1$ glues have type "0", followed by r glues of type "1", r glues of type "2", and so on. The exposed south, east, and west tile edges have glues g_S, g_E, and g_W, respectively.

In order to build the flexible decompression pad, a modified decompression pad representing a number $C = c_0c_1\ldots c_{d-1}$ in base 2^d is needed.

▶ **Lemma 16.** *Given an integer $d \geq 3$, there exists a staged assembly system with 1 bin, $8d-1$ tile types, and 1 stage that assembles a width-3 length-d^2 flexible decompression pad.*

Proof. Start with the construction of Lemma 7 that yields a a width-2 length-d^2 decompression pad encoding C. Modify the tile types of this construction such that the leftmost northmost glue is s' and every non-s glue on the north surface is a special *prime* version, to differentiate between other similar glue types. Then add $2d+1$ tiles that modify the north surface decompression pad to yield width-3 flexible decompression pad, as seen in Figure 8.

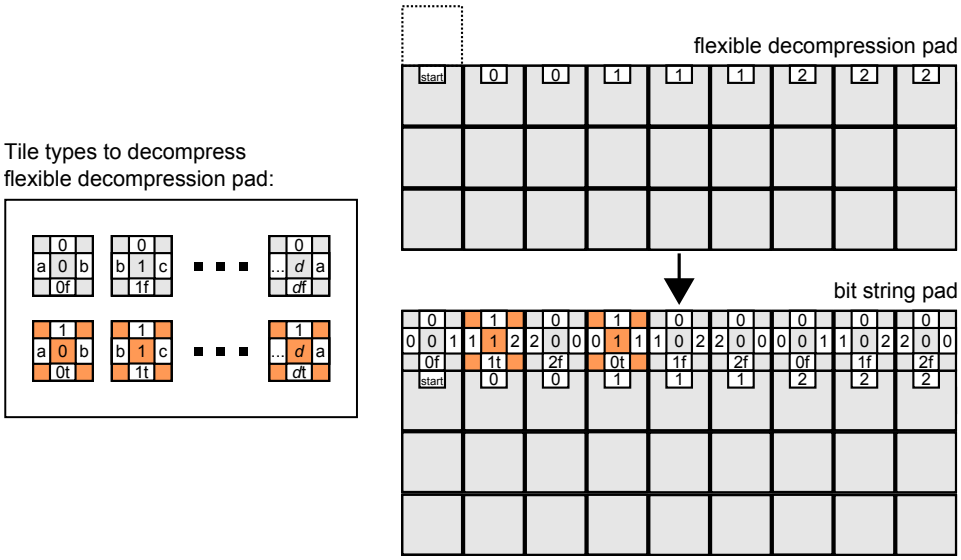

Figure 9 On the left, the templates for the decompression tiles needed to decompress a flexible decompression pad for any given $d \geq 3$. In the top right, an example of a length-9 flexible decompression pad. In the bottom right, the decompression tiles interact with the flexible decompression pad and glue function to assemble a bit string pad from a flexible decompression pad, representing the bit string "010100000". The flexible glues form a bond of strength 2 between the glue pair ('start', '0f'), strength 1 between glues pairs ('0', '0'), ('1', '1'), ('2', '2'), ('0', '1t'), ('0', '2f'), ('1', '0t'), ('1', '1f'), ('1', '2f'), ('2', '0f'), ('2', '1f'), and ('2', '2f'), and strength 0 between all other glue pairs.

This step requires $5d + \log 2^d - 2$ tile types to build a modified decompression pad and $2d + 1$ tiles to convert this modified decompression pad that into a flexible decompression pad. Thus $2d + 1 + 5d + \log 2^d - 2 \leq 8d - 1$ tile types are used in total. ◀

▶ **Lemma 17.** *Given integers $d \geq 3$ and any length d^2 bit string R, there exists a 1 stage, 1 bin, staged assembly system with flexible glues that assembles a width-4 gap-0 d^2-bit string pad representing R, using at most $10d - 1$ tile types.*

Proof. Consider a width-3, length d^2 flexible decompression pad. The idea is to use $2d$ tile types and flexible glues to build a width-4 gap-0 d^2-bit string pad from the flexible decompression pad (see Figure 9). Consider a sequence of d bitstrings $D = D_0, D_1, \ldots, D_{d-1}$ with each $D_i = s_0 s_1 s_2 \ldots s_{d-1}$ such that the in-order concatenation of all bitstrings in D equals R. Let $D_{i,j}$ denote the j^{th} bit of the i^{th} bit string of D.

The goal is to construct a glue function such that it specifies the tiles that can attached to the top of the flexible decompression pad to be the concatenation of the bitstrings in D. Note that the tiles that have a "0" or "1" glue as those with labels that end in "f" or "t", respectively. Let $str(g_1, g_2)$ denote the strength between glues g_1 and g_2. Set the tile that attaches to the "start" glue to be one that exposes "0" or a "1" by setting $str(start, 0f) = 2$ or $str(start, 0t) = 2$, respectively. For all $D_{i,j}$, we set $str(i, jf) = 0$ if and only if $D_{i,j} = 0$ and $str(i, jt) = 1$, otherwise. In addition, we set $str(a, a) = 1$, $str(b, b) = 1$, and so on.

With this, we build a width-4 gap-0 d^2-bit string pad from the flexible decompression pad. An example of this can be seen in Figure 9. Also, $8d - 1$ tile types are used to build a width-3, length d^2 flexible decompression pad. An additional $2d$ tile types are needed to decompress, using flexible glues, into a width-4 d^2-bit string pad. So the total number of tile types used is $10d - 1$. ◀

▶ **Lemma 18.** *Given $t \in \mathbb{N}$, there exists some constant $c \in \mathbb{N}$, such that for all cases where $t \geq c$, there exists a staged self-assembly system with t tiles which assembles any width-4 $\Theta(t^2)$-bit string pad using 1 stage, 1 bin, and flexible glues.*

Proof. Given t tile types, consider how many bits can be produced using Lemma 17. Let $d = \lfloor \frac{t+1}{10} \rfloor$. Invoke Lemma 17 to build a width-4 d^2-bit string pad with flexible glues using $10d-1$ tiles. The number of bits produced is $y = d^2 = (\lfloor \frac{t+1}{10} \rfloor)^2 = \Theta(t^2)$. Then by Lemma 17, any width-4 $(\lfloor \frac{t+1}{10} \rfloor)^2$-bit string pad can be build in the flexible glue model using at most t tiles, 1 stage, and 1 bin. The smallest choice of d requires $d = \lfloor \frac{t+1}{10} \rfloor \geq 3$, implying $t \geq 29$. For all cases where $t \geq c$ we have a constant, $c = 29$, where this lemma holds true. ◀

The improvements to Lemmas 17 and 18 allow for a larger bit string pad to be built in Step 2 when compared to standard glues, reducing stage complexity to $\mathcal{O}(\frac{\log\log b}{\log t} + \frac{x-tb-t^2}{b^2})$:

▶ **Lemma 19.** *Given $t, b \in \mathbb{N}$ and a bit string r where $x = |r|$. Then, there exist some constants $c, d \in \mathbb{N}$, such that for all cases where $t > c$ and $b > d$, there is a staged self-assembly system using flexible glues with b bins and t tiles which assembles an x-bit string pad representing r with width 9 and gap $\Theta(\log b)$ using $\mathcal{O}(\frac{x-tb-t^2}{b^2} + \frac{\log\log b}{\log t})$ stages.*

Nearly tight upper and lower bounds for square and general shape construction in the flexible glue model are obtained by replacing the bit string construction of Lemma 3 with Lemma 19, and applying the flexible glue lower bound of Lemma 1:

▶ **Theorem 20.** *For any $b, t, n \in \mathbb{N}$ and constants c_b, c_t such that $b \geq c_b$ and $t \geq c_t$, there exists a staged self-assembly system using flexible glues with b bins and t tile types that uniquely produces an $n \times n$ square using $\mathcal{O}(\frac{\log n - t^2 - tb}{b^2} + \frac{\log\log b}{\log t})$ stages.*

▶ **Theorem 21.** *For any $b, t \in \mathbb{N}$ and almost all $n \in \mathbb{N}$, any staged self-assembly system with flexible glues which uses at most b bins and t tile types that uniquely assembles an $n \times n$ square must use $\Omega(\frac{\log n - t^2 - tb}{b^2})$ stages.*

▶ **Theorem 22.** *For any shape S and $b, t \in \mathbb{N}$ and constants c_b, c_t such that $b \geq c_b$ and $t \geq c_t$, there exists a staged self-assembly system using flexible glues with b bins and t tile types which uniquely produces S at some scale factor using $\mathcal{O}(\frac{K(S) - t^2 - tb}{b^2} + \frac{\log\log b}{\log t})$ stages.*

▶ **Theorem 23.** *For any $b, t \in \mathbb{N}$ and shape S with Kolmogorov complexity $K(S)$, any staged self-assembly system with flexible glues which uses at most b bins and t tile types that uniquely assembles S must use $\Omega(\frac{K(S) - t^2 - tb}{b^2})$ stages.*

8 Conclusion

In this work, we achieved nearly optimal staged assembly of two classic benchmark shape classes. These constructions generalize the known upper bounds of [2, 6, 8, 20] to arbitrary choices of tile type and bin counts, as well as to the flexible glue model. The natural problem left open is the elimination of the additive $\mathcal{O}(\frac{\log\log b}{\log t})$ gap between the upper and lower bounds induced by the wings subconstruction of Section 4.1. Although this subconstruction is the cause of an additive gap in an otherwise optimal result, it is a useful approach for general assembly labeling and coordinated attachment and is likely useful in other staged constructions. The constant width of our bit string pads can also potentially be exploited for efficient construction of shapes with geometric bottlenecks, e.g., thin rectangles.

References

1. Zachary Abel, Nadia Benbernou, Mirela Damian, Erik Demaine, Martin Demaine, Robin Flatland, Scott Kominers, and Robert Schweller. Shape replication through self-assembly and RNAse enzymes. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2010.
2. Leonard Adleman, Qi Cheng, Ashish Goel, and Ming-Deh Huang. Running time and program size for self-assembled squares. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 740–748, 2001.
3. Bahar Behsaz, Ján Maňuch, and Ladislav Stacho. Turing universality of step-wise and stage assembly at temperature 1. In *DNA Computing and Molecular Programming (DNA)*, volume 7433 of *LNCS*, pages 1–11. Springer, 2012.
4. Sarah Cannon, Erik D. Demaine, Martin L. Demaine, Sarah Eisenstat, Matthew J. Patitz, Robert Schweller, Scott M. Summers, and Andrew Winslow. Two hands are better than one (up to constant factors): Self-assembly in the 2HAM vs. aTAM. In *Proceedings of 30th International Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 20 of *LIPIcs*, pages 172–184. Schloss Dagstuhl, 2013.
5. Ho-Lin Chen and David Doty. Parallelism and time in hierarchical self-assembly. In *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1163–1182, 2012.
6. 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*, 34:1493–1515, 2005.
7. E. D. Demaine, M. J. Patitz, T. A. Rogers, R. T. Schweller, and D. Woods. The two-handed tile assembly model is not intrinsically universal. In *Automata, Languages and Programming (ICALP)*, volume 7965 of *LNCS*, pages 400–412. Springer, 2013.
8. 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(3):347–370, 2008.
9. Erik D. Demaine, Sarah Eisenstat, Mashhood Ishaque, and Andrew Winslow. One-dimensional staged self-assembly. *Natural Computing*, 12(2):247–258, 2013.
10. Erik D. Demaine, Sándor P. Fekete, Christian Scheffer, and Arne Schmidt. New geometric algorithms for fully connected staged self-assembly. In *DNA Computing and Molecular Programming (DNA)*, volume 9211 of *LNCS*, pages 104–116. Springer, 2015.
11. 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). In *Proceedings of the 28th International Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 9 of *LIPIcs*, pages 201–212. Schloss Dagstuhl, 2011.
12. David Doty. Producibility in hierarchical self-assembly. In *Proceedings of Unconventional Computation and Natural Computation (UCNC) 2014*, pages 142–154, 2014.
13. Constantine Evans. *Crystals that Count! Physical Principles and Experimental Investigations of DNA Tile Self-Assembly.* PhD thesis, Caltech, 2014.
14. David Furcy, Samuel Micka, and Scott M. Summers. Optimal program-size complexity for self-assembly at temperature 1 in 3D. In *DNA Computing and Molecular Programming (DNA)*, volume 9211 of *LNCS*, pages 71–86. Springer, 2015.
15. Yonggang Ke, Luvena L. Ong, William M. Shih, and Peng Yin. Three-dimensional structures self-assembled from dna bricks. *Science*, 338(6111):1177–1183, 2012.
16. Thomas H. Labean, Sung Ha Park, Sang Jung Ahn, and John H. Reif. Stepwise DNA self-assembly of fixed-size nanostructures. In *Foundations of Nanoscience, Self-assembled Architectures, and Devices*, pages 179–181, 2005.

17 Ján Maňuch, Ladislav Stacho, and Christine Stoll. Step-wise tile assembly with a constant number of tile types. *Natural Computing*, 11(3):535–550, 2012.

18 Matthew J. Patitz and Scott M. Summers. Identifying shapes using self-assembly. *Algorithmica*, 64:481–510, 2012.

19 Paul W. K. Rothemund and Erik Winfree. The program-size complexity of self-assembled squares (extended abstract). In *Proceedings of the 32nd ACM Symposium on Theory of Computing (STOC)*, pages 459–468, 2000.

20 David Soloveichik and Erik Winfree. Complexity of self-assembled shapes. *SIAM Journal on Computing*, 36(6):1544–1569, 2007.

21 Erik Winfree. *Algorithmic Self-Assembly of DNA*. PhD thesis, Caltech, 1998.

22 Andrew Winslow. Staged self-assembly and polyomino context-free grammars. *Natural Computing*, 14(2):293–302, 2015.