Algorithmica manuscript No.
(will be inserted by the editor)

Optimal Staged Self-Assembly of General Shapes*

Cameron Chalk · Eric Martinez · Robert Schweller · Luis Vega · Andrew Winslow · Tim Wylie

Received: date / Accepted: date

Abstract We analyze the number of tile types t, bins b, and stages necessary to assemble $n \times n$ squares and scaled shapes in the staged tile assembly model. For $n \times n$ squares, we prove $\mathcal{O}(\frac{\log n - tb - t\log t}{b^2} + \frac{\log\log b}{\log t})$ stages suffice and $\Omega(\frac{\log n - tb - t\log t}{b^2})$ are necessary for almost all n. For shapes S with Kolmogorov complexity $K(S)$, we prove $\mathcal{O}(\frac{K(S) - tb - t\log t}{b^2} + \frac{\log\log b}{\log t})$ stages suffice

* Research supported in part by National Science Foundation Grants CCF-1117672, CCF-1555626, and CCF-1422152. An abstract version of this work appeared as [5].

C. Chalk
University of Texas Rio Grande Valley
Department of Computer Science
E-mail: cameron.chalk01@utrgv.edu

E. Martinez
University of Texas Rio Grande Valley
Department of Computer Science
E-mail: eric.m.martinez02@utrgv.edu

R. Schweller
University of Texas Rio Grande Valley
Department of Computer Science
E-mail: robert.schweller@utrgv.edu

Luis Vega
University of Texas Rio Grande Valley
Department of Computer Science
E-mail: luis.a.vega01@utrgv.edu

Andrew Winslow
Previously at Université Libre de Bruxelles
University of Texas Rio Grande Valley
Department of Computer Science
E-mail: andrew.winslow@utrgv.edu

T. Wylie
University of Texas Rio Grande Valley
Department of Computer Science
E-mail: timothy.wylie@utrgv.edu

and $\Omega(\frac{K(S)-tb-t\log t}{b^2})$ are necessary to assemble a scaled version of S, for almost all S. We obtain similarly tight bounds when the more powerful *flexible* glues are permitted.

Keywords DNA computing · biocomputing · staging · 2HAM · hierarchical

1 Introduction

Self-assembly is the process by which many small, simple particles attach via local interactions to form large, complex structures. In the early 1980s, Ned Seeman [24] demonstrated that short strands of DNA, attaching via matching base pairs, could be "programmed" to carry out such self-assembly at the nanoscale. In the 1990s, the Ph.D. thesis of Erik Winfree [26] introduced an approach for DNA self-assembly wherein four-sided DNA "tiles" attach edgewise via matching sequences of base pairs.

Such *DNA tile self-assembly* is both experimentally [15] and theoretically [26, 23] capable of complex behaviors. Study of theoretical models of such tile assembly, beginning with the *abstract Tile Assembly Model (aTAM)* of Winfree [26], has grown into a field with dozens of model variations [13, 21] compared via a web of reductions [29], reminiscent of the development of structural complexity for traditional models of computation.

Staged tile assembly. The *staged (tile assembly) model* is a generalization of the *two-handed (2HAM)* [1, 4, 11, 8] or *hierarchical* [6, 14] *tile assembly model*. In the aTAM, single tiles attach to a growing multi-tile assembly. The 2HAM additionally permits multi-tile assemblies to attach to each other, yielding a growth process strictly more general than that of the aTAM [4].

The staged assembly model extends the 2HAM by including the ability to simultaneously carry out separate assembly processes in multiple *bins*. Such separation is motivated by similar experimental practices wherein many reactions are carried out simultaneously in distinct test tubes, or bins, often automated via liquid-handling robots [17].

In the staged model, a bin begins with a set of *input assemblies* previously assembled in other bins. These assemblies are repeatedly attached pairwise to yield a growing set of *producible assemblies* until all possibilities are exhausted. The producible assemblies not attachable to any other producible assemblies during this process are the *output assemblies* of the bin, and may be used as input assemblies for other bins.

An instance of the staged model, called a *staged system*, consists of many bins stratified into stages, and a *mix graph* that specifies which bins in each stage supply input assemblies for bins in the following stage. Input assemblies for bins in the first stage are sets of individual tiles, and the output assemblies of bins in the final stage are considered the *output* of the system.

Complexity in staged assembly. A common goal in the design of self-assembling systems is the assembly of a desired shape. Here we consider the design of "efficient" systems with minimum complexity that assemble a given shape. Three metrics exist for staged systems:

- *Tile type complexity:* the number of distinct tile types used in the system.
- *Bin complexity:* the maximum number of bins used in a stage.
- *Stage complexity:* the number of stages.

Efficient assembly of restricted classes of shapes [8,10] and $1 \times n$ "patterned lines" [9,27] have been considered, and efficient assembly in other variants and further extensions and variants of the staged self-assembly model have also been studied [1,3,12,18,20,22].

Two classes of shapes have become the defacto standards [28] for measuring assembly efficiency: squares and general shapes (with scaling permitted). Efficient assembly of these two classes was first considered in the aTAM, where matching upper and lower bounds on the tile complexity were obtained [23, 25, 2].

Our results. Here we give nearly matching upper and lower (trivariate) bounds for assembling these shapes in the staged model; our results are summarized here and in Table 1. Due to the multivariate nature of the results, they are described using two complexity measures (tile type and bin) as "independent" variables and the third measure (stage) as the "dependent" variable. That is, the results are upper and lower bounds on the minimum stage complexity of systems with a fixed number of tile types t and bins b, where t and b are restricted to be larger than some fixed constant.[1]

For $n \times n$ squares, we prove the stage complexity is $\mathcal{O}(\frac{\log n - tb - t\log t}{b^2} + \frac{\log\log b}{\log t})$ and, for almost all n, $\Omega(\frac{\log n - tb - t\log t}{b^2})$.[2] For shapes S with Kolmogorov complexity[3] $K(S)$, we prove the stage complexity is $\mathcal{O}(\frac{K(S) - tb - t\log t}{b^2} + \frac{\log\log b}{\log t})$ and $\Omega(\frac{K(S) - tb - t\log t}{b^2})$.

We obtain similar results when *flexible glues* [7], glues that can form bonds with non-matching glues, are permitted. In this case, the stage complexity for $n \times n$ squares is reduced to $\mathcal{O}(\frac{\log n - t^2 - tb}{b^2} + \frac{\log\log b}{\log t})$ and, for almost all n, $\Omega(\frac{\log n - t^2 - tb}{b^2})$, and the stage complexity for general shapes S with Kolmogorov complexity $K(S)$ is reduced to $\mathcal{O}(\frac{K(S) - t^2 - tb}{b^2} + \frac{\log\log b}{\log t})$ 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 desired 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 efficiency. In other words, the number of bits in the bit string pad grows proportionally to the number of bits needed to describe the system.

Comparison with prior work. Because our results are optimal across all choices of tile type and bin complexity, these results generalize and, in some cases, improve on prior results. For instance, Theorem 1 implies assembly 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

[1] Such a restriction is necessary, as systems with a single tile type are incapable of assembling finite non-trivial shapes.

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

[3] For more information on Kolmogorov complexity, we suggest [19].

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 [7].

The same theorem also yields assembly of $n \times n$ squares by systems with $\mathcal{O}(1)$ bins, $\mathcal{O}(1)$ tile types, and $\mathcal{O}(\log n)$ stages (a result of [8]) and by systems with $\mathcal{O}(\sqrt{\log n})$ 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 3 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 [25].

Standard Glue Stage Complexity Results

Shape	Upper Bound	Lower Bound	Theorems
$n \times n$	$\mathcal{O}(\frac{\log n - t \log t - tb}{b^2} + \frac{\log \log b}{\log t})$	$\Omega(\frac{\log n - t \log t - tb}{b^2})$	1,2
Scaled shapes	$\mathcal{O}(\frac{K(S) - t \log t - tb}{b^2} + \frac{\log \log b}{\log t})$	$\Omega(\frac{K(S) - t \log t - tb}{b^2})$	3,4

Flexible Glue Stage Complexity Results

$n \times n$	$\mathcal{O}(\frac{\log n - t^2 - tb}{b^2} + \frac{\log \log b}{\log t})$	$\Omega(\frac{\log n - t^2 - tb}{b^2})$	6
Scaled shapes	$\mathcal{O}(\frac{K(S) - t^2 - tb}{b^2} + \frac{\log \log b}{\log t})$	$\Omega(\frac{K(S) - t^2 - tb}{b^2})$	7,8

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.

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 $\mathbf{u} = \langle u_x, u_y \rangle \in \mathbb{Z}^2$, $A + \mathbf{u}$ 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 $B = A + \mathbf{u}$ for some vector $\mathbf{u}$. For a configuration A, the *assembly* of A is the set $\tilde{A} = \{B : B \simeq A\}$. An assembly $\tilde{A}$ is a *subassembly* of an assembly $\tilde{B}$, denoted $\tilde{A} \sqsubseteq \tilde{B}$, provided that there exists an $A \in \tilde{A}$ and $B \in \tilde{B}$ such that $A \subseteq B$. 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 $A \bigcup B = C$, $\text{dom(A)} \bigcap \text{dom(B)} = \emptyset$, 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 upper bounds, and at most some constant for lower bounds. 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 $C \in P'_{(S,\tau)}$.

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 assemblies that can assemble beginning from the initial set S, whereas $P_{(S,\tau)}$ represents only the set of assemblies that cannot grow any further.

The assemblies in $P_{(S,\tau)}$ are *uniquely produced* iff for each $x \in P'_{(S,\tau)}$ there exists a $y \in P_{(S,\tau)}$ such that $x \sqsubseteq y$. Unique production implies that every non-terminal 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.[4] The *output* of a staged system is the union of the set of terminal assemblies of the bins in the final stage.[5] The output of a staged

[4] 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.

[5] 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 it is considered for lower bound purposes. However, all results herein apply to both variants of the model.

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 proves an upper bound on the information content of a staged system, and thus a lower bound on the stage complexity for assembling shapes containing a specified quantity of information. 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.*

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. 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 almost all bit strings, any staged system with b bins and t tiles that uniquely produces a terminal assembly encoding 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, where x is the length of the bit string.

Our positive results rely mainly on efficient assembly of *bit string pads*: assemblies that expose a sequence of north-facing glues that encode a bit string. An example is shown in Figure 1, on the left. Squares and general scaled shapes are assembled by combining a universal set of "computation" tiles with efficiently assembled "input" bit string pads.

Definition 1 (Bit string pad) A *width-k gap-f r-bit string pad* is a $k \times (f(r-1)+1)$ rectangular assembly (throughout this work we describe rectangular assemblies by vertical *width* $\times$ horizontal *length*) with r glues from a set of two glue types $\{g_0, g_1\}$ exposed on the north face of the rectangle at

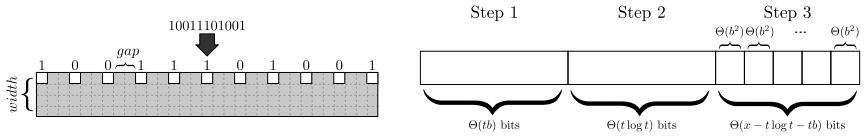

Fig. 1 For compactness, glues are denoted by their subscripts in figures for the remainder of the paper (e.g. g_0 denoted as 0). Left: an example bit string $r = 10011101001$ encoded as a width-4 gap-2 11-*bit string pad* whose north-facing glues correspond to the bits in r. Right: the decomposition of an x-bit string pad's bits into those encoded by the three steps of a staged system with t tile types and b bins.

intervals of length f, starting from the westmost northern edge. All remaining exposed glues on the north tile edges have some common label g_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 g_0 and g_1 glues from west to east, on the north facing edges, are equal to the given bit string.

For our upper bounds we efficiently construct bit string pads by decomposing the target pad into three subpads and assembling each in a separate step using a different source of system complexity (see Figure 1(b)):

- Step 1: $\Theta(tb)$ bits from assigning tile types to stage-1 bins (Section 4.3).
- Step 2: $\Theta(t \log t)$ bits from the tile types themselves as in [2,7,16,25] (Section 4.4).
- Step 3: $\Theta(x - t \log t - tb)$ bits from the mix graph using a variant of "crazy mixing" [8] (Section 4.5).

If flexible glues are permitted, Step 2 is modified as in [7] to achieve $\Theta(t^2)$ bits from tile types. These subpads are then combined into the complete pad to achieve the following result:

Lemma 2 *There exists a constant c such that for any $b, t \in \mathbb{N}$ with $b, t > c$ and any bit string R of length x, there exists a $\tau = 2$ staged assembly system with b bins, t tiles, and $\mathcal{O}(\frac{x-tb-t\log t}{b^2} + \frac{\log\log b}{\log t})$ stages whose uniquely produced output is a width-7 gap-$\Theta(\log b)$ x-bit string pad representing R.*

Proof Consider a bit string R of length x. The width-7 gap-$\Theta(\log b)$ x-bit string pad representing R to be constructed is referred to as the *target bit string pad*, and the staged assembly system which assembles this pad is the *target system*. The techniques used in the target system require that the numbers of tile types and bins must be greater than a constant c. The target system is designed by allotting a constant fraction of the system's total tile types and bins (i.e. $\frac{t}{c}$ and $\frac{b}{c}$ for some constant c) to each of three steps. The three steps each assemble a subpad of the target bit string pad, which are then concatenated. The steps' subpads and stage complexities are as follows:

- Step 1: A width-7 gap-$\Theta(\log b)$ $\Theta(tb)$-bit string pad representing $\Theta(tb)$ bits of R. Uses $\mathcal{O}(\frac{\log\log b}{\log t})$ stages (see Lemma 4).

- Step 2: A width-3 gap-$\Theta(\log b)$ $\Theta(t \log t)$-bit string pad representing $\Theta(t \log t)$ bits of R (see Lemma 7). Uses $\mathcal{O}(\frac{\log \log b}{\log t})$ stages to induce the $\Theta(\log b)$ gap as discussed in Section 4.4.1.
- Step 3: A width-7 gap-$\Theta(\log b)$ $(x - \Theta(tb) - \Theta(t \log t))$-bit string pad representing the remaining $(x - \Theta(tb) - \Theta(t \log t))$ bits of R. Uses $\mathcal{O}(\frac{x - tb - t \log t}{b^2} + \frac{\log \log b}{\log t})$ stages (see Lemma 8).

The subpads constructed by these steps cannot simply be concatenated, as their adjacent bits at the point of concatenation would not have the same uniform gap as the other bits. A length-$\Theta(\log b)$ *connector* must be assembled to concatenate the subpads while inducing the appropriate gap. Lemma 3, although describing a system which assembles an assembly with more requirements than needed for this purpose, proves the existence of a system with t tile types and b bins which can assemble a length-$\Theta(\log b)$ assembly using $\mathcal{O}(\frac{\log \log b}{\log t})$ stages. Such a system can trivially be modified to expose glues on east and west ends for use in concatenating the subpads. In one stage, concatenate two bit string pads using a copy of this connector assembly, and in one more stage concatenate the third. $\mathcal{O}(1)$ tile types, bins and stages can be used to "fill in" the portions of the assembly with width less than 7.

Steps 1, 2, and the concatenation of the three subpads each use $\mathcal{O}(\frac{\log \log b}{\log t})$ stages, whereas Step 3 uses $\mathcal{O}(\frac{x - tb - t \log t}{b^2} + \frac{\log \log b}{\log t})$ stages. Then the total number of stages used by the target system is $\mathcal{O}(\frac{x - tb - t \log t}{b^2} + \frac{\log \log b}{\log t})$. □

The additive gap between the upper and lower bounds implied by these lemmas is due to the $\mathcal{O}(\frac{\log \log b}{\log t})$ additional stages used to assemble some of the machinery needed to carry out the three steps of Lemma 2. We leave the removal of this gap as an open problem, stated in Section 8.

4 Bit String Pad Construction

As sketched in Section 3, we assemble bit string pads from three subpads constructed via distinct methods utilizing distinct sources of information complexity in staged self-assembly systems. The construction is described in five parts constituting the remainder of this section. The first two parts, Sections 4.1 and 4.2, describe helpful subconstructions used by the three subpad constructions that follow in Sections 4.3 through 4.5.

4.1 Wings

In this section, a b bin system constructs all possible $\lfloor \log b \rfloor$-bit string pads, each contained in its own bin. The purpose of these bit string pads is to mix them with $\mathcal{O}(1)$ tile types to assemble *wings*. Wings are rectangular assemblies with geometric bumps, or *teeth*, that encode a positive integer *index* in binary. A wing gadget has index i and m bits provided it geometrically encodes an

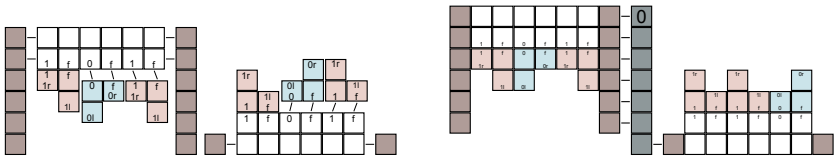

Fig. 2 Assembling left and right wings from bit string pads and extra $\mathcal{O}(1)$-size assemblies.

m-bit binary string representing i. Wings come in two varieties, *west* and *east*, shown in Figure 2.

Wings encode their indices on either the north surface (west wings) or south surface (east wings). Roughly speaking, a pair of west and east wings can attach if their teeth match (perfectly interlock), equivalent to having the same index. In Steps 1 and 3, west and east wings are attached to the west and east ends of *bit sticks*, which are 7×1 assemblies which expose a north-facing g_0 or g_1 glue. The attached wings direct these bit sticks to assemble in a desired west-to-east order as seen in Figure 3. The bit string pads underlying the wings are constructed as follows:

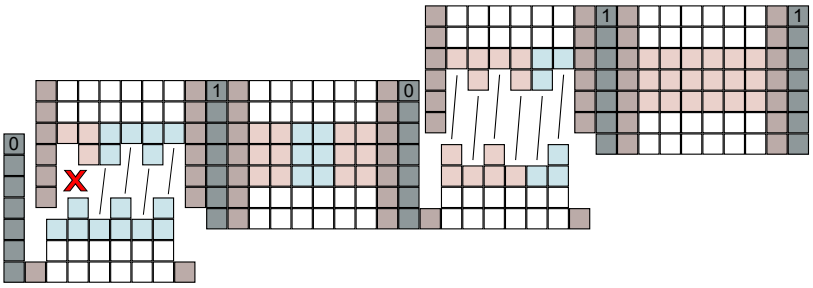

Fig. 3 Directing the assembly of bit sticks using wings. At left, mismatched wings are prevented from attaching due to mismatched geometry while at right, matching wings have no such geometric mismatch and so attach.

Lemma 3 *There exists a constant c such that for any $b, t \in \mathbb{N}$ with $b, t > c$, there exists a $\tau = 2$ staged assembly system with b bins, t tile types, and $\mathcal{O}(\frac{\log \log b}{\log t})$ stages whose uniquely produced output is all width-2 gap-1 $\lfloor \log(b) \rfloor$-bit string pads, each placed in a distinct bin.*

Proof Let $\gamma = \lfloor \frac{t-4}{2} \rfloor + 1$. The construction has three cases, depending upon the relative values of γ (determined by t) and b. Intuitively, Cases 1 and 2 start with gap-0 1-bit string pads and *connectors* are mixed in to concatenate these bit string pads to yield larger gap-1 bit string pads representing more bits. This process is repeated to yield all width-2 gap-1 $\lfloor \log(b) \rfloor$-bit string pads.

Case 1: $\gamma < 2 \log b$ and $b = \gamma^n$ for $n \in \mathbb{N}$. All three cases use width-2, gap-0 1-bit string pads called *binary gadgets*: $\mathcal{O}(1)$-sized assemblies representing 0 and 1. The remaining tile types are used to assemble $2(\gamma - 1)$ *connectors* that

bind to the west and east ends of binary gadgets. Connectors come in *west* and *east* varieties and each connector has a positive integer *index* such that two connectors can bind to one another if and only if their indices are equal.

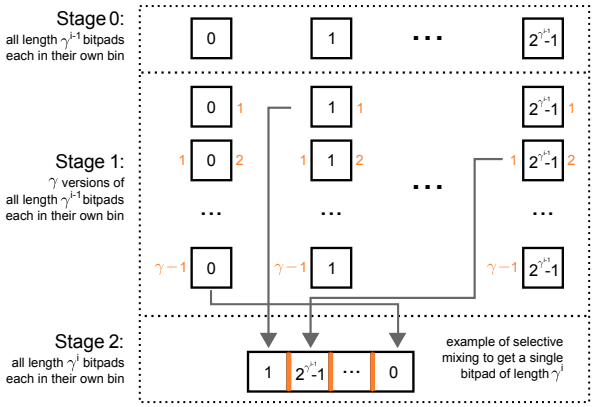

Fig. 4 The 2-stage Round i used in Case 1 of Lemma 3. Stage 0 contains bit string pads representing $0, 1, \ldots, 2^{\gamma^{i-1}} - 1$. Stage 1 is roughly similar to Stage 0, only with γ versions of the bit string pads pads from Stage 0. These γ versions differ only by the presence of connectors on the west and east ends of the bit string pads. From Stage 1 to Stage 2, γ-sized subsets of these modified bit string pads are mixed to yield larger bit string pads. Intuitively, the output of Stage 2 is similar to Stage 0 but yielding bit string pads with a factor of γ more bits, allowing another round to begin, if necessary.

For each $k \in \{1, 2, \ldots, \gamma-1\}$, assemble west and east connectors with index k. Then repeat the following 2-stage "round" seen in Figure 4. The ith round begins with $2^{\gamma^{i-1}}$ bit string pads, each in their own bin, and each of the west and east connectors stored in their own bins. In the first stage of the round, mix each bit string pad with west and east connectors with indices $h-1$ and h, respectively, for all $h \in \{1, 2, \ldots, \gamma-1\}$. In the special case that $h = 1$ or $h = \gamma - 1$, omit the west or east connector, respectively. The result is $\gamma 2^{\gamma^i}$ distinct bit string pads (with attached connectors), with each bit string pad in its own bin. The tile set for bit string pads and connectors can be seen in Figure 5.

In the second stage of the round, selectively mix the γ-size subsets of the $\gamma 2^{\gamma^{i-1}}$ bins, choosing one bit string pad with each pair of connector indices, to assemble all width-2 γ^i-bit string pads in separate bins. The number of bins used in round i is thus $2^{\gamma^i} + \gamma$: enough for all γ^i-bit string pads and the γ (reusable) connectors.

Perform sufficient rounds to assemble all $\lfloor \log(b) \rfloor$-bit string pads, i.e., the smallest integer r such that $\gamma^r \geq \log(b)$ and thus $r = \lfloor \log_\gamma \log(b) \rfloor$. In the last stage of round r, the number of bins used can be reduced to 2^{γ^r}, dropping the γ additional bins containing connectors.

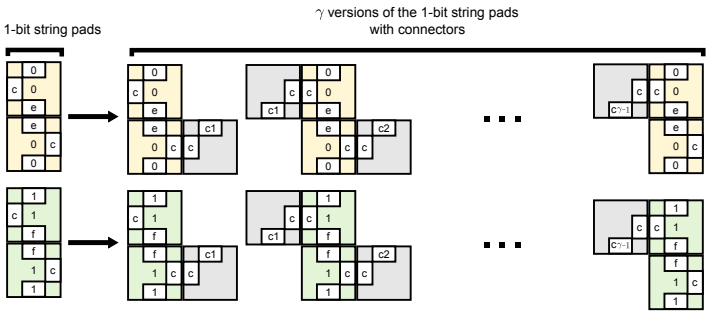

Fig. 5 Left: 2 width-2 1-bit string pads. Right: 2γ versions of each bit string pad with pairs of west and east connectors attached, as done in the first stage of each round.

The number of bins used then does not exceed b since $2^{\gamma^i} + \gamma \leq 2^{\gamma^r} \leq b$ for all $i < r$. Moreover, the number of stages used is $\mathcal{O}(r) = \mathcal{O}(\frac{\log\log b}{\log \gamma}) = \mathcal{O}(\frac{\log\log b}{\log t})$. This construction requires $b, t > c$ for some constant c large enough to ensure $\gamma \geq 1$ and b can accommodate at least 1 of each of the bin types.

Case 2: $\gamma < 2\log(b)$ and $b \neq \gamma^n$ for $n \in \mathbb{N}$. Note that this case is identical to Case 1, except that b is not a power of γ. Thus the desired length of the assembled bit string pads are between γ^{r-1} and γ^r for some r; as a solution, bit string pads are assembled from collections of shorter bit string pads of power-of-γ lengths.

Let $d_{\lfloor \log_\gamma(b) \rfloor} d_{\lfloor \log_\gamma(b) \rfloor - 1} \ldots d_2 d_1$ be the base-γ expansion of b. New *special* connectors with index 0 are used, in addition to the *standard* connectors from Case 1. Each round also uses *growth* bins, *output* bins, and *west* and *east incubator* bins. West and east incubator bins contain growing $\sum_{j=1}^{i} d_j \gamma^{j-1}$-bit string pads with special west or east connectors, respectively.

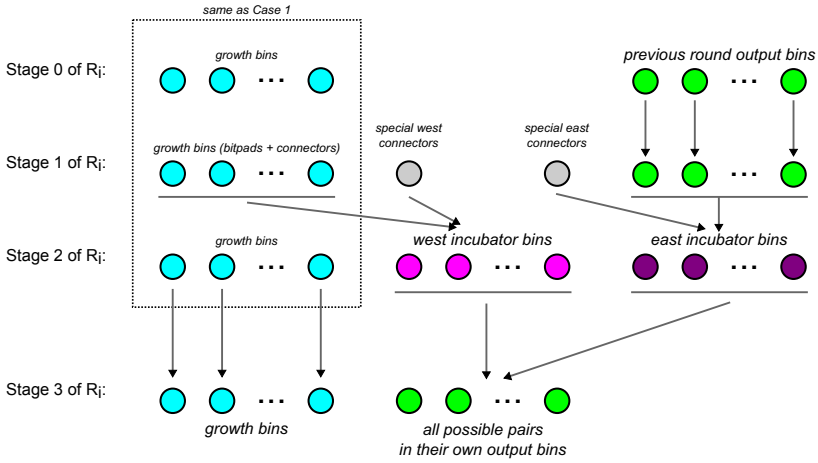

Fig. 6 The 3-stage Round i used in Case 2 of Lemma 3.

As in Case 1, use $\log_\gamma(b)$ rounds to assemble growing sets of longer bit string pads. The ith round begins with all γ^{i-1}-bit string pads in separate growth bins. In the first stage of the ith round, mix standard connectors with these pads to assemble γ versions of all γ^{i-1}-bit string pads in separate growth bins. Also, the output bins are equal to their predecessors in the previous stage.

In the second stage of the ith round, if $d_i > 0$, then selectively mix γ-sized subsets of bit string pads with standard connectors (stored in growth bins) to assemble all $d_i\gamma^{i-1}$-bit string pads in separate bins. If $i = 1$ or the output bins are empty, these bit string pads are stored in their own output bins.

Otherwise, the output bins are not empty; in this case they are also mixed with west connectors and stored in west incubator bins. In the same stage, mix every bit string pad from the output bins with a special east connector and store them in east incubator bins.

In the third stage, mix all pairs of west and east incubator bins into separate output bins, replacing any previous output bins.

For all but the last round, mix γ-sized subsets of bit string pads with standard connectors (stored in growth bins) to assemble all γ^i-bit string pads in separate growth bins, replacing the previous growth bins. Since each round concatenates $d_i\gamma^{i-1}$ bits to the bit string pads in output bins, the bit string pads assembled in the output bins of the final round represent $\sum_{j=1}^{\lfloor \log_\gamma(b) \rfloor} d_j\gamma^{j-1} = \lfloor \log(b) \rfloor$ bits. The high-level idea for the construction can be seen in Figure 6.

This construction requires $b, t > c$ for some constant c large enough to ensure that $\gamma \geq 1$ and at least one of each bin type (growth, output, etc.) is available. The number of stages remains $\mathcal{O}(\frac{\log\log b}{\log t})$.

Case 3: $\gamma \geq 2\log(b)$**.** Let $\beta = \lfloor \log(b) \rfloor$. Begin with β bins, each containing a binary gadget representing 0 with a distinct pair of west/east glues g_1 and g_2, g_2 and g_3, etc. Similarly, create a second set of identical β bins, but with binary gadgets representing 1. In the second stage, combine every β-sized subset of bins that contain binary gadgets with distinct west/east glue pairs to assemble all β-bit string pads. □

4.2 Fattening

Roughly speaking, *fattening* replaces a single tile type with a set of 5 corresponding tile types and a *filler* assembly of length $k-2$ to yield a $2 \times k$ *fattened* assembly which exposes the same glues as the original tile (see Figure 7). The filler assembly determines the length of the fattened assembly. Fattening is used as a subroutine in Steps 1 and 2 to increase the gap of the subpads they assemble from 0 to $\Theta(\log b)$, matching that of the subpad assembled by Step 3, and as a subroutine in Step 2 for assembling decompression pads. The filler assembly required for fattening the Step 1 and 2 subpad tile sets should be of length $\Theta(\log b)$. Lemma 3 yields all length-$\log(b)$ bit string pads in $\mathcal{O}(\frac{\log\log b}{\log t})$ stages, which implies a method to construct one length-$\log(b)$ assembly in the same stage complexity.

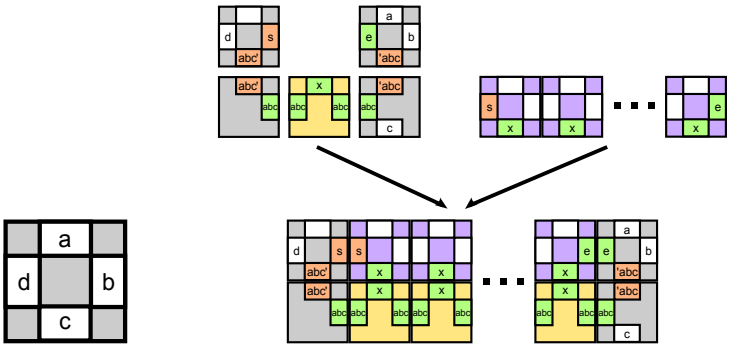

Fig. 7 Left: the tile to be fattened. Right: 5 tile types, based on the tile to be fattened, are mixed with a length-$(k-2)$ filler assembly which "fattens" the assembly from width 1 to width k. The four glues of the original are exposed in the same directions.

4.3 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 with 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 - dominated by the use of the wings subconstruction of Section 4.1. The assignment yields $\Theta(b)$ bins that contain subpads encoding distinct length-$\Theta(t)$ substrings of the $\Theta(tb)$ bits. These assemblies are then combined in the proper order using wings.

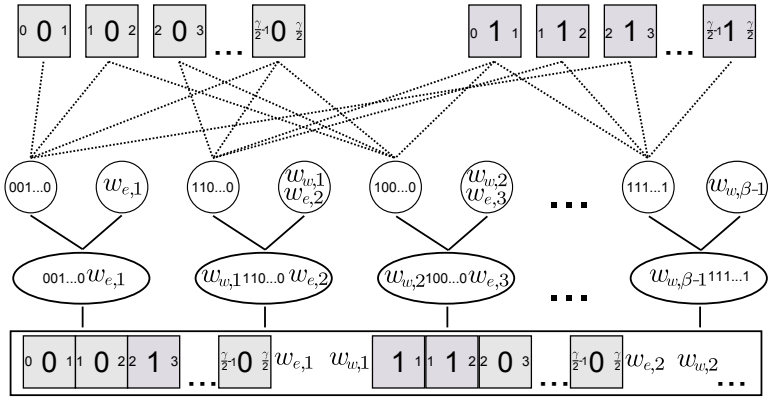

Fig. 8 The assembly of a $\frac{\gamma\beta}{2}$-bit string pad. γ and β equate (asymptotically) to the system's number of tile types and bins respectively. The squares labeled 0 and 1 represent bit sticks. The dotted lines indicate tile to bin assignments before the first stage of the system. This assignment allows specific $\frac{\gamma}{2}$-bit string pads to be combined, one for each of β bins. Then, these pads are concatenated in the desired order using wings; $w_{e,i}$ and $w_{w,i}$ represent the ith east and west wings respectively.

Lemma 4 *There exists a constant c such that for any $b, t \in \mathbb{N}$ with $b, t > c$, there exists an $x = \Theta(tb)$ such that for any bit string R of length x, there exists a $\tau = 2$ staged assembly system with b bins, t tile types, and $\mathcal{O}(\frac{\log\log b}{\log t})$ stages whose uniquely produced output is a width-7 gap-$\Theta(\log b)$ x-bit string pad representing R.*

Proof See Figure 8 for a sketch of the idea. Let γ and β be constant fractions of t and b, respectively, and b, t large enough such that $\gamma, \beta \geq 1$. Use γ tiles and β bins to construct all west and east $\log(\beta)$-bit wings according to Section 4.1. Also construct γ distinct bit sticks, $\frac{\gamma}{2}$ representing 0 and $\frac{\gamma}{2}$ representing 1. Additionally, the bit sticks' east and west sides expose glues such that any $\frac{\gamma}{2}$-bit string pad can be assembled from $\frac{\gamma}{2}$ bit sticks attached sequentially.

Each bit string pad constructed this way must also be *fattened* using the technique described in Section 4.2. Using $\mathcal{O}(\frac{\log\log\beta}{\log\gamma})$ stages, each pad is fattened to length g where g is the length of a wing. The purpose of fattening is to ensure the complete bit string pad constructed has uniform gap g.

In each of β bins, assemble $\frac{\gamma}{2}$ bit sticks into a distinct $\frac{\gamma}{2}$-bit string subpad of the desired pad. Combine each of these β length-$\frac{\gamma}{2}$ subpads with wings that encode their sequential ordering in the pad (i.e. the wings are used to ensure that the subpads attach in the desired order). Then, assemble these "wing-labeled" subpads into the complete $\frac{\gamma\beta}{2} = \Theta(tb)$-bit string pad. The resultant bit string pad will have gap-g, since each bit in each subpad has been fattened to length g and wings of length g are used to concatenate each subpad.

This gap satisfies $g = \Theta(\log b)$, since the wings used are 1-gapped $\log\beta$-bit string pads with $\mathcal{O}(1)$ additional length added (as seen in Figure 2). Particularly, $g = 2(\log\beta) + 2$. The number of stages used is $\mathcal{O}(\frac{\log\log\beta}{\log\gamma}) = \mathcal{O}(\frac{\log\log b}{\log t})$ (for the wings, see Lemma 3) plus $\mathcal{O}(1)$ (the subpads of the desired pad). □

4.4 Step 2: encoding via tile types

Here the goal is to design a collection of t tile types that assembles a target string of $\Theta(t\log t)$ bits in bit string pad form. The solution is to utilize a common base conversion approach in tile assembly used by [2,7,16,25]. In this approach, tile types optimally encode integer values in a large base and are then "decompressed" into a binary representation. In total, t tile types are used to encode a value in a high base and decompress this representation into a string of $\Theta(t\log t)$ bits.

Definition 2 (Decompression pad) For $k, d, x \in \mathbb{N}$ and $u = 2^x$, a width-k d-digit base-u decompression pad is a $k \times dx$ rectangular assembly with d glues from a set of $u-1$ glue types $\{g_0, g_1, ..., g_{u-1}\}$ exposed on the north face of the rectangle at intervals of length $x-1$ and starting from the westmost northern edge. All remaining glues on the north surface have a common type g_N. The remaining exposed south, east, and west tile edges have glues g_S, g_E, and g_W. The exposed glues on the northern edge, disregarding glues of type g_N, $g_{a_1}, g_{a_2}, \ldots, g_{a_d}$ represent string $a_1, a_2, \ldots, a_d$.

The base conversion approach is illustrated by an example: assembling a width-3 9-bit string pad representing $R = 010100000$ ($R = 240$ in base 8). First, a decompression pad representing R in base 8 by combining 3 different $2 \times \log_2(8)$ blocks is assembled (see Figure 9). Then, this decompression pad is combined with $\mathcal{O}(u)$ tile types to "decompress" the 3-digit base-8 representation of R into an 8-bit base-2 representation (see Figure 10).

The remainder of this section consists of three lemmas. The first, Lemma 5, establishes that decompression pads representing d-digit base-u strings can be assembled with $\mathcal{O}(d + \log(u))$ tile types. The second, Lemma 6, describes the decompression tile types used to "decompress" the high-base representation on the decompression pad. The third, Lemma 7, optimizes the choice of base used to minimize the number of tile types used in total for both the decompression pad and decompression tiles.

The resulting bit string pads are gap-0; the follow-up Section 4.4.1 describes how to increase the gap of these bit string pads to match those assembled in Steps 1 and 3 by applying fattening to a subset of the tile types used.

Lemma 5 *Given integers $x \geq 3$, $d \geq 1$ and $u = 2^x$, there exists a $\tau = 2$ staged self-assembly with 1 bin, at most $5d + \log(u) - 2$ tile types, and 1 stage system that uniquely produces a width-2 d-digit base-u decompression pad.*

Proof Consider a number $R = s_0 s_1 \ldots s_{d-1}$ in a base u. The goal is to build a *decompression pad* with north glues representing the d digits of R, and $\log(u) - 1$ spacing between consecutive digits exposing north glues g_B. As depicted in Figure 9, a width-2 length-$\log(u)$ decompression pad is built for each of the d digits in the base-u representation of R. These d length-$\log(u)$ decompression pads then assemble into the complete length-$d\log(u)$ decompression pad representing R.

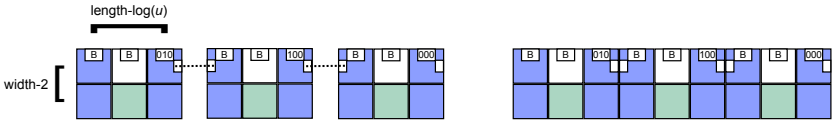

Fig. 9 Assembling a decompression pad representing a bit string $R = 010\,100\,000$ in base $u = 8$. Left: three width-2 length-$\log(u)$ decompression pads are designed for each digit in the base-u representation of R. Right: the decompression pad assembled into a width-2 length-$|R|$ decompression pad representing R in base u.

The decompression pads for each digit require 5 unique tile types along with a *shared filler* of length $\log(u) - 2$ built with $\log(u) - 2$ tile types. The shared filler is an optimization that allows a portion of each length-$\log(u)$ decompression pad to be assembled with some shared tile types. Therefore, the total tile complexity is $5d + \log(u) - 2$. □

Lemma 6 *Given integers $d \geq 3$, $x \geq 3$, $u = 2^x$, and a $d\log(u)$-bit number R, there exists a $\tau = 2$ staged assembly system with 1 bin, $5d + 2u + \log(u) - 4$ tile*

types, and 1 stage that uniquely produces a width-3 gap-0 $d\log(u)$-bit string pad representing R.

Proof Since R is a $d\log(u)$-bit number, it can be represented as a d-digit number in base u. Use Lemma 5 to obtain a tile set that assembles into a decompression pad that represents R in base u (requiring d digits and thus $5d + \log(u) - 2$ tile types).

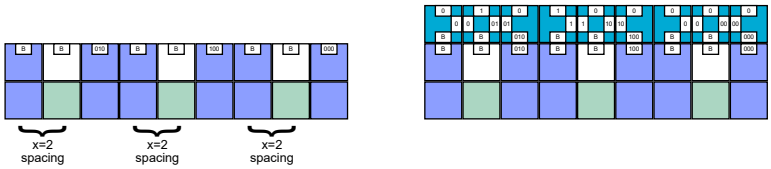

Fig. 10 Left: a width-2 gap-$(\log(u) - 1)$ decompression pad representing a bit string $R = 010100000$ in base $u = 8$. Right: $\mathcal{O}(u)$ decompression tiles interact with the north glues of the decompression pad to combine into a width-3 bit string pad representing R in base 2.

Use an additional $2u - 2$ tile types to "decompress" the high-base representation of R by exposing a single bit via north glues and passing along the remaining bits of the digit via east and west glues (see Figure 11).

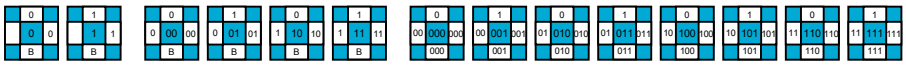

Fig. 11 The set of tiles used to decompress a decompression pad representing a string in base 8. Since $u = 8$, $2u - 2 = 14$ tile types are used. These tiles attach to the exposed glues on a decompression pad, replacing a large-gap high-base representation of R with a 0-gap low-base representation.

In total, $\log(u)$ attachments are used to unpack the base-u digit into a sequence of $\log(u)$ bits. The total tile complexity is $t = 5d + 2u + \log(u) - 4$: $5d + \log(u) - 2$ types to build the decompression pad and $2u - 2$ decompression tile types. □

Lemma 7 *There exists a constant c such that for any $t \in \mathbb{N}$ with $t > c$, there exists an $x = \Theta(t \log t)$ such that for any bit string R of length x, there exists a $\tau = 2$ staged assembly system with 1 bin, t tile types, and 1 stage that uniquely assembles the width-3 gap-0 x-bit string pad representing R.*

Proof Here, the large base u used in Lemma 6 is optimized to maximize the number of bits encoded by the construction. Recall that Lemma 6 first constructs length-$\log(u)$ decompression pads for every digit in a base-u string. Decompression pads are constructed using 5 tile types plus some additional number of shared "spacing" tile types that determine the gap length of the decompression pad. Then ignoring spacing tile types, $\lfloor \frac{t}{2} \rfloor$ tile types assemble $\lfloor \frac{t}{10} \rfloor$ decompression pads and so $\lfloor \frac{t}{10} \rfloor$ base-u digits can be represented.

Next, Lemma 6 uses $2u + \log(u)$ tile types for "decompressing" the decompression pads and shared spacing. Since $\lfloor \frac{t}{2} \rfloor$ tile types were used to assemble decompression pads, only $\lfloor \frac{t}{2} \rfloor$ available tile types remain and thus $2u + \log(u) \leq \lfloor \frac{t}{2} \rfloor$.

The number of bits x encoded by the assembled bit string pad is $x = d \log(u)$. The choice of $u = 2^{\lfloor \log \frac{t}{3} \rfloor}$ satisfies the aforementioned bound on tile types while (asymptotically) maximizing u. The result is a bit string pad with $x = d \log(u) = \lfloor \frac{t}{10} \rfloor \log 2^{\lfloor \log \frac{t}{3} \rfloor} = \lfloor \frac{t}{10} \rfloor \lfloor \log \frac{t}{3} \rfloor = \Theta(t \log t)$ bits.

Due to constraints building the decompression pad, the high-base string must have at least 3 digits and base at least 8. For building the decompression pad, minus the shared spacing, at least $\lfloor \frac{t}{2} \rfloor \geq 5d = 15$ tile types are needed. For the decompression tiles and shared spacing, at least $\lfloor \frac{t}{2} \rfloor \geq 2u + \log(u) - 4 = 15$ tile types are needed. For base 8, $u = 2^{\lfloor \log \frac{t}{3} \rfloor} = 8$. So this optimization requires $\frac{t}{2} \geq 15$ and $\frac{t}{3} \geq 8$ and thus $t \geq c = 30$. □

4.4.1 Gapped bit string pads

The subpads assembled by Sections 4.3 and 4.5 are gap-$\Theta(\log b)$, in contrast with the gap-0 subpad assembled here. As a solution, we increase the gap of these subpads from 0 to $\Theta(\log b)$ using a modified version of the fattening process described in Section 4.2.

Let $q = \Theta(\log b)$ be the length of the desired gap. In Lemma 7, width-2 decompression pads representing strings of base-u digits are assembled. To increase the gap, a portion of the tile types used to assemble these decompression pads are replaced with fattened versions to space out the g_B glues with g_f glues, resulting in a width-3 decompression pad. Figure 12 shows an example of the modification.

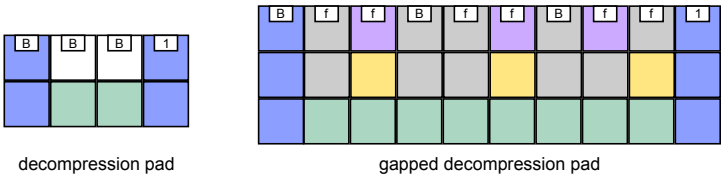

Fig. 12 A gap-0 width-2 decompression pad (left) and a gap-2 width-3 decompression pad (right). Increasing the gap to non-zero values requires only constant additional tile types per decompression pad and slight modifications to the glues.

4.5 Step 3: encoding via mix graph

In this step, a system is designed wherein the mix graph is carefully designed to choose bits by mixing particular bins, a technique termed *crazy mixing* [8]. Observe that there are b^2 potential mix graph edges between a pair of adjacent

stages of a b-bin staged assembly system. Thus any choice of edges can be specified using b^2 bits - one bit per edge.

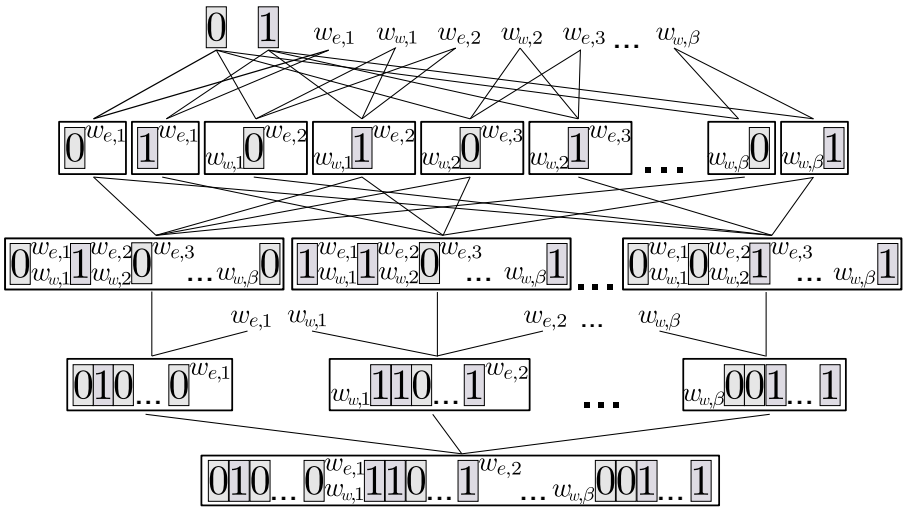

Fig. 13 The assembly of β^2-bit string pads using β wings and $\mathcal{O}(1)$ stages. The rectangles 0 and 1 represent bit sticks that may attach wings on either side; $w_{e,i}$ and $w_{w,i}$ represent the ith east and west wings respectively.

Lemma 8 *There exists a constant c such that for any $b, t \in \mathbb{N}$ with $b, t > c$ and any bit string R of length x, there is a $\tau = 2$ staged assembly system with b bins, t tile types, and $\mathcal{O}(\frac{x}{b^2} + \frac{\log\log b}{\log t})$ stages that uniquely assembles a width-7 gap-$\Theta(\log b)$ x-bit string pad representing R.*

Proof The construction divides the target x-bit string pad into b^2-bit subpads, and a $\mathcal{O}(1)$-stage process is repeated to specify the b^2 bits of each of these subpad. An overview of the construction is shown in Figure 13.

Let γ and β represent some constant fractions of t and b respectively. Utilize γ tiles and β bins to construct all length-$\log_2(\beta)$ west and east wings according to Section 4.1. Denote the ith west and east wings by $w_{w,i}$ and $w_{e,i}$, respectively. Also construct two *bit sticks*: width-7, gap-0 1-bit sting pads, i.e. 7×1 assemblies that expose 1 or 0 north glues.

Stage 1. Mix $w_{e,i}$ and $w_{w,i-1}$ with bit stick 0 into a bin labeled b_i^0 for all $1 \leq i \leq \beta$. Similarly, mix $w_{e,i}$ and $w_{w,i-1}$ with bit stick 1 into a bin labeled b_i^1. In total, 2β bins are used.

Stage 2. Selectively mix specific bit sticks 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.

Stage 3. Attach wings to each of the β-bit string pads. For each of the β bins, mix $w_{e,i}$ and $w_{w,i-1}$ into the bins such that $w_{e,1}$ is mixed with the first β bits of the desired β^2-bit string pad, $w_{e,2}$ and $w_{w,1}$ are mixed with the second β bits of the desired β^2-bit string pad, etc.

Stage 4. 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, attaching each β^2-bit string pad onto a growing assembly containing all previous pads, to yield the final x-bit string pad. In total, $\mathcal{O}(\frac{\log\log\beta}{\log\gamma})$ 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 the total stage complexity is $\mathcal{O}(\frac{x}{\beta^2} + \frac{\log\log\beta}{\log\gamma}) = \mathcal{O}(\frac{x}{b^2} + \frac{\log\log b}{\log t})$. □

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 [23]. 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's), the assembly stops growing. The length of the resulting rectangle is determined by the seed bit string pad on which binary counting begins. Two rectangles assembled this way serve as two adjacent sides of the desired square. The square can be filled in using these two rectangles as binding locations for generic filler tiles. 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 1 *There exists a constant c such that for any $b, t, n \in \mathbb{N}$ with $b, t > c$, there exists a $\tau = 2$ staged 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 whose uniquely produced output is an assembly whose shape is 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 [23]. 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 2, 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 14 shows the construction, including modifications to the technique shown in [23].

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

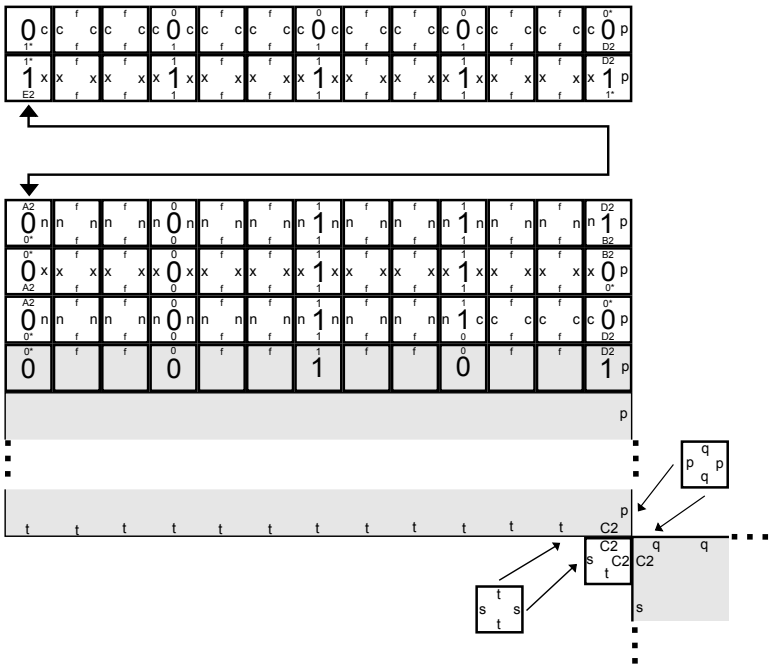

Fig. 14 Constructing a counter seed. The bit string pads are shown in gray. Glues with a "2" in the string have strength-2, all other glues have strength 1.

as in [23] 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 southmost eastern (northmost western) 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 [23] 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 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 2 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 2 *For any $b, t \in \mathbb{N}$ and almost all $n \in \mathbb{N}$, any staged assembly system which uses at most b bins and t tile types whose uniquely produced output is an assembly whose shape is 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 [25]. 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.

Theorem 3 *There exists a constant c such that for any $b, t \in \mathbb{N}$ with $b, t > c$ and any shape S of Kolmogorov complexity $K(S)$, there exists a $\tau = 2$ staged 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 whose uniquely produced output has shape S at some scale factor.*

Proof Observe that the tile set described in [25] 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 [25] to carry out the simulation of a (fixed) universal Turing machine. Let $b' = \frac{b-1}{5}$.

Use the method of Lemma 2 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 1 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 15. The Turing simulation occurs in one stage by mixing the four combined bit string pads into one bin that contains the fixed set of tile types simulating a Turing machine described in [25]. The bit string pads contain spacing between the exposed binary glues, while the simulation tile

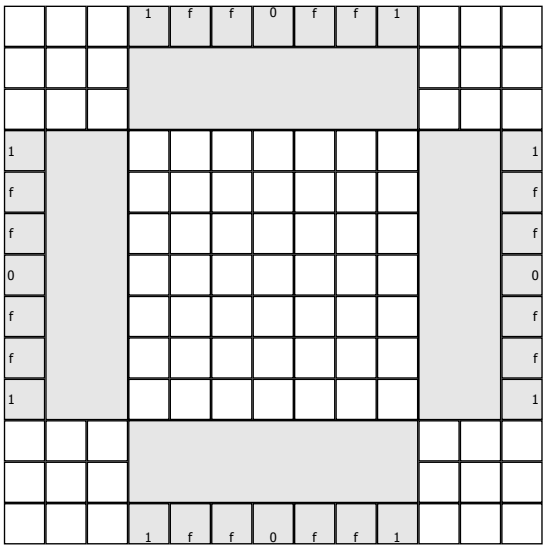

Fig. 15 The modified seed block used in scaled shape assembly. Bit string pads are colored in gray. We combine four $K(S)$-bit string pads into a square, each representing a Kolmogorov-optimal description of S.

types of [25] 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 1. 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.

Theorem 4 *For any $b, t \in \mathbb{N}$ and shape S with Kolmogorov complexity $K(S)$, any staged assembly system which uses at most b bins and t tile types whose uniquely produced output has shape S must use $\Omega(\frac{K(S)-t \log t - tb}{b^2})$ stages.*

7 Flexible Glues

Here, we consider reducing stage complexity using *flexible glues* that allow non-equal glue pairs to have positive strength. These can be used to encode $\Theta(t^2)$ bits rather than $\Theta(t \log t)$ bits in t tile types, reducing both the upper and lower stage complexity bounds. For the upper bound, the additional bits are encoded by modifying Step 2 of the bit string pad construction of Section 4 to use a modified decompression pad, similar to the technique introduced in [7].

Definition 3 (Flexible decompression pad) A width-k length-d^2 flexible decompression pad is a $k \times d^2$ rectangular assembly with d^2 north glue types from the set $\{g_{\text{start}}, g_0, g_1, \ldots, g_d\}$ exposed on the north face of the rectangle. The westmost glue is g_{start}, the following $d-1$ glues have type g_0, followed by d glues of type g_1, d glues of type g_2, and so on. The exposed south, east, and west tile edges have glues g_S, g_E, and g_W, respectively.

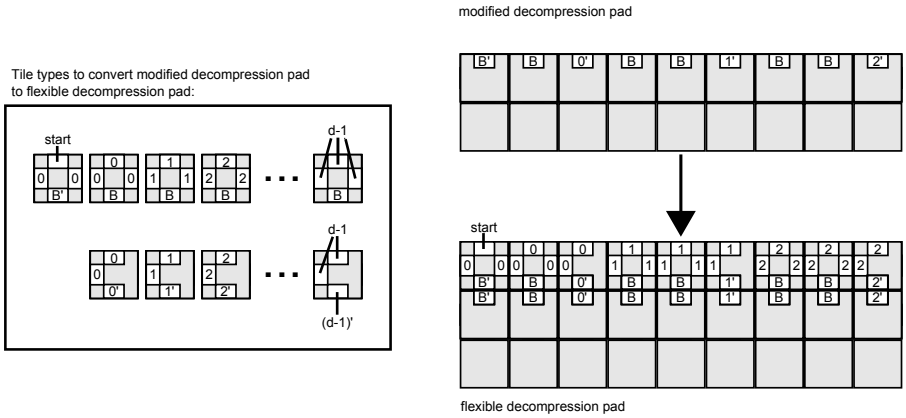

Fig. 16 The templates to convert a modified decompression pad to a flexible decompression pad using $2d+1$ tile types, where integer $d \geq 3$, on the left. Using these additional tile types, a modified decompression pad is converted into a flexible decompression pad. A *modified decompression pad* has a westmost northmost glue of $g_{B'}$ and every non-g_B glue on the north surface is a special *prime* version distinct from other similar glue types. On the right, a width-2 modified decompression pad representing the string 012 in base-8 is converted to a width-3 length-9 flexible decompression pad.

In order to build the flexible decompression pad, a modified decompression pad representing a number $C = c_0 c_1 \ldots c_{d-1}$ in base 2^d is needed. The construction of the flexible decompression pad described in Lemma 9 allows flexible glues to interact on the north surface of the flexible decompression pad to yield a bit string pad, as described in Lemma 10, and using fewer tile types than the non-flexible glue construction by encoding information in the glue interactions.

Lemma 9 *Given an integer $d \geq 3$, there exists a $\tau = 2$ 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 5 that yields a width-2 length-d^2 decompression pad encoding C using $5d + \log 2^d - 2$ tile types. Modify the tile types of this construction such that the westmost northern glue is $g_{B'}$ and every non-g_B glue on the north surface is a special *prime* version, to differentiate between other similar glue types. Then add $2d+1$ tile types that modify the north surface decompression pad to yield width-3 flexible

decompression pad, as seen in Figure 16. This step requires $5d + \log 2^d - 2$ tile types to build a modified decompression pad and $2d + 1$ tile types to convert this modified decompression pad into a flexible decompression pad. Thus $2d + 1 + 5d + \log 2^d - 2 \leq 8d - 1$ tile types are used in total. □

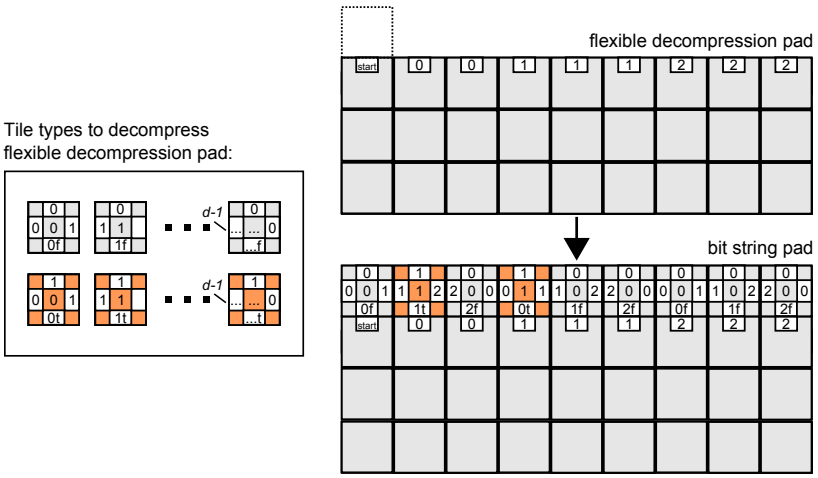

Fig. 17 On the left, the templates for the decompression tiles needed to decompress a flexible decompression pad for any given $d \geq 3$. In the top right, an example of a length-9 flexible decompression pad. In the bottom right, the decompression tiles interact with the flexible decompression pad and 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 (g_{start}, g_{0f}), strength 1 between glues pairs (g_0, g_0), (g_1, g_1), (g_2, g_2), (g_0, g_{1t}), (g_0, g_{2f}), (g_1, g_{0t}), (g_1, g_{1f}), (g_1, g_{2f}), (g_2, g_{0f}), (g_2, g_{1f}), and (g_2, g_{2f}), and strength 0 between all other glue pairs.

Lemma 10 *Given an integer $d \geq 3$ and any length d^2 bit string R, there exists a $\tau = 2$ staged assembly system with flexible glues with 1 bin, at most $10d - 1$ tile types, and 1 stage whose uniquely produced output is a width-4 gap-0 d^2-bit string pad representing R.*

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 17). 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 jth bit of the ith bit string of D.

The goal is to construct a glue function such that it specifies the tiles that can attach 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" label as those with glues that end in "f" or "t", respectively. Let $str(g, g')$ denote the strength between any two glues g and g'. Set the tile that attaches to the g_{start} glue to be

one that exposes g_0 or a g_1 by setting $str(g_{\text{start}}, g_{0f}) = 2$ or $str(g_{\text{start}}, g_{0t}) = 2$, respectively. For all $D_{i,j}$, we set $str(g_i, g_{jf}) = 1$ if and only if $D_{i,j} = 0$ and $str(g_i, g_{jt}) = 1$, otherwise. In addition, we set $str(g_0, g_0) = 1$, $str(g_1, g_1) = 1$, and so on, up to $str(g_{d-1}, g_{d-1}) = 1$.

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 17. 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 11 *There exists a constant c such that for any $t \in \mathbb{N}$ with $t > c$ there exists an $x = \Theta(t^2)$ such that for any bit string R of length x, there exists a $\tau = 2$ staged assembly system with flexible glues with 1 bin, t tile types and 1 stage whose uniquely produced output is a width-4 gap-0 x-bit string pad representing R.*

Proof Given t tile types, consider how many bits can be produced using Lemma 10. Let $d = \lfloor \frac{t+1}{10} \rfloor$. Invoke Lemma 10 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 10, any width-4 $(\lfloor \frac{t+1}{10} \rfloor)^2$-bit string pad can be built 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 10 and 11 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 12 *There exists a constant c such that for any $b, t \in \mathbb{N}$ with $b, t > c$ and a bit string R of length x, there is a $\tau = 2$ staged assembly system with flexible glues with b bins, t tile types, and $\mathcal{O}(\frac{x - tb - t^2}{b^2} + \frac{\log \log b}{\log t})$ stages whose uniquely produced output is a width-9 gap-$\Theta(\log b)$ x-bit string pad representing R.*

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 2 with Lemma 12, and applying the flexible glue lower bound of Lemma 1:

Theorem 5 *There exists a constant c such that for any $b, t, n \in \mathbb{N}$ such that $b, t > c$, there exists a $\tau = 2$ staged assembly system with flexible glues with b bins, t tile types and $\mathcal{O}(\frac{\log n - t^2 - tb}{b^2} + \frac{\log \log b}{\log t})$ stages whose uniquely produced output is an $n \times n$ square.*

Theorem 6 *For any $b, t \in \mathbb{N}$ and almost all $n \in \mathbb{N}$, any staged assembly system with flexible glues which uses at most b bins and t tile types whose uniquely produced output is an $n \times n$ square must use $\Omega(\frac{\log n - t^2 - tb}{b^2})$ stages.*

Theorem 7 *There exists a constant c such that for any shape S and $b, t \in \mathbb{N}$ with $b, t > c$, there exists a $\tau = 2$ staged assembly system with flexible glues with b bins, t tile types, and $\mathcal{O}(\frac{K(S)-t^2-tb}{b^2} + \frac{\log\log b}{\log t})$ stages whose uniquely produced output is S at some scale factor.*

Theorem 8 *For any $b, t \in \mathbb{N}$ and shape S with Kolmogorov complexity $K(S)$, any staged assembly system with flexible glues which uses at most b bins and t tile types whose uniquely produced output is S must use $\Omega(\frac{K(S)-t^2-tb}{b^2})$ stages.*

8 Conclusion

In this work, we achieve nearly optimal staged assembly of two classic benchmark shape classes. These constructions generalize the known upper bounds of [2,7,8,25] to all combinations of tile type and bin complexity, as well as to the flexible glue model.

The obvious technical question that remains is whether the additive $\mathcal{O}(\frac{\log\log b}{\log t})$ gap between the upper and lower bounds can be removed. This gap in our constructions is induced by the wings subconstruction of Section 4.1, and seems difficult to eliminate, as the wings serve as our generic solution to assembly labeling and coordinated attachment. As such, the wings subconstruction might be useful for improving the efficiency of staged assembly for other shape classes.

Finally, the constant width of the bit string pads we construct was not exploited here, but may be useful for assembling shapes with geometric bottlenecks, e.g., thin rectangles.

References

1. Abel, Z., Benbernou, N., Damian, M., Demaine, E.D., Demaine, M.L., Flatland, R., Kominers, S., Schweller, R.: Shape replication through self-assembly and RNAse enzymes. In: Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA) (2010)
2. Adleman, L., Cheng, Q., Goel, A., Huang, M.D.: Running time and program size for self-assembled squares. In: Proceedings of the 33rd Annual ACM Symposium on Theory of Computing (STOC), pp. 740–748 (2001)
3. Behsaz, B., Maňuch, J., Stacho, L.: Turing universality of step-wise and stage assembly at temperature 1. In: DNA Computing and Molecular Programming (DNA), *LNCS*, vol. 7433, pp. 1–11. Springer (2012)
4. Cannon, S., Demaine, E.D., Demaine, M.L., Eisenstat, S., Patitz, M.J., Schweller, R., Summers, S.M., Winslow, A.: 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), *LIPIcs*, vol. 20, pp. 172–184. Schloss Dagstuhl (2013)
5. Chalk, C., Martinez, E., Schweller, R., Vega, L., Winslow, A., Wylie, T.: Optimal staged self-assembly of general shapes. In: Proceedings of the 24th Annual European Symposium on Algorithms (ESA), *LIPIcs*, vol. 57, pp. 26:1–26:17. Schloss Dagstuhl (2016)
6. Chen, H.L., Doty, D.: Parallelism and time in hierarchical self-assembly. In: Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 1163–1182 (2012)

7. Cheng, Q., Aggarwal, G., Goldwasser, M.H., Kao, M.Y., Schweller, R.T., de Espanés, P.M.: Complexities for generalized models of self-assembly. SIAM Journal on Computing **34**, 1493–1515 (2005)
8. Demaine, E.D., Demaine, M.L., Fekete, S.P., Ishaque, M., Rafalin, E., Schweller, R.T., Souvaine, D.L.: Staged self-assembly: nanomanufacture of arbitrary shapes with $O(1)$ glues. Natural Computing **7**(3), 347–370 (2008)
9. Demaine, E.D., Eisenstat, S., Ishaque, M., Winslow, A.: One-dimensional staged self-assembly. Natural Computing **12**(2), 247–258 (2013)
10. Demaine, E.D., Fekete, S.P., Scheffer, C., Schmidt, A.: New geometric algorithms for fully connected staged self-assembly. In: DNA Computing and Molecular Programming (DNA), *LNCS*, vol. 9211, pp. 104–116. Springer (2015)
11. Demaine, E.D., Patitz, M.J., Rogers, T.A., Schweller, R.T., Woods, D.: The two-handed tile assembly model is not intrinsically universal. In: Automata, Languages and Programming (ICALP), *LNCS*, vol. 7965, pp. 400–412. Springer (2013)
12. Demaine, E.D., Patitz, M.J., Schweller, R.T., Summers, S.M.: 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), *LIPIcs*, vol. 9, pp. 201–212. Schloss Dagstuhl (2011)
13. Doty, D.: Theory of algorithmic self-assembly. Communications of the ACM **55**(12), 78–88 (2012)
14. Doty, D.: Producibility in hierarchical self-assembly. Natural Computing **15**(1), 41–49 (2016)
15. Evans, C.: Crystals that count! physical principles and experimental investigations of dna tile self-assembly. Ph.D. thesis, Caltech (2014)
16. Furcy, D., Micka, S., Summers, S.M.: Optimal program-size complexity for self-assembly at temperature 1 in 3D. In: DNA Computing and Molecular Programming (DNA), *LNCS*, vol. 9211, pp. 71–86. Springer (2015)
17. Ke, Y., Ong, L.L., Shih, W.M., Yin, P.: Three-dimensional structures self-assembled from dna bricks. Science **338**(6111), 1177–1183 (2012)
18. Labean, T.H., Park, S.H., Ahn, S.J., Reif, J.H.: Stepwise DNA self-assembly of fixed-size nanostructures. In: Foundations of Nanoscience, Self-assembled Architectures, and Devices, pp. 179–181 (2005)
19. Li, M., Vitnyi, P.M.: An Introduction to Kolmogorov Complexity and Its Applications, 3 edn. Springer Publishing Company, Incorporated (2008)
20. Maňuch, J., Stacho, L., Stoll, C.: Step-wise tile assembly with a constant number of tile types. Natural Computing **11**(3), 535–550 (2012)
21. Patitz, M.J.: An introduction to tile-based self-assembly and a survey of recent results. Natural Computing **13**(2), 195–224 (2014)
22. Patitz, M.J., Summers, S.M.: Identifying shapes using self-assembly. Algorithmica **64**, 481–510 (2012)
23. Rothemund, P.W.K., Winfree, E.: The program-size complexity of self-assembled squares (extended abstract). In: Proceedings of the 32nd ACM Symposium on Theory of Computing (STOC), pp. 459–468 (2000)
24. Seeman, N.C.: Nucleic-acid junctions and lattices. Journal of Theoretical Biology **99**, 237–247 (1982)
25. Soloveichik, D., Winfree, E.: Complexity of self-assembled shapes. SIAM Journal on Computing **36**(6), 1544–1569 (2007)
26. Winfree, E.: Algorithmic self-assembly of DNA. Ph.D. thesis, Caltech (1998)
27. Winslow, A.: Staged self-assembly and polyomino context-free grammars. Natural Computing **14**(2), 293–302 (2015)
28. Winslow, A.: A brief tour of theoretical tile self-assembly. In: Proceedings of the 22nd International Workshop on Cellular Automata and Discrete Complex Systems (AUTOMATA), *LNCS*, vol. 9664, pp. 26–31. Springer (2016)
29. Woods, D.: Intrinsic universality and the computational power of self-assembly. Philosophical Transaction of the Royal Society A **373**(2046) (2015)