Randomized Self-Assembly for Exact Shapes

David Doty
Department of Computer Science
Iowa State University
Ames, IA 50011, USA
ddoty@iastate.edu

Abstract— Working in Winfree's abstract tile assembly model, we show that a constant-size tile assembly system can be programmed through relative tile concentrations to build an $n \times n$ square with high probability, for any sufficiently large n. This answers an open question of Kao and Schweller (*Randomized Self-Assembly for Approximate Shapes*, ICALP 2008), who showed how to build an *approximately* $n \times n$ square using tile concentration programming, and asked whether the approximation could be made *exact* with high probability.

1. Introduction

Self-assembly is a term used to describe systems in which a small number of simple components, each following local rules governing their interaction with each other, automatically assemble to form a target structure. Winfree [21] introduced the abstract Tile Assembly Model (aTAM) – based on a constructive version of Wang tiling [19], [20] – as a simplified mathematical model of Seeman's work [15] in utilizing DNA to physically implement self-assembly at the molecular level. In the aTAM, the fundamental components are un-rotatable, translatable square "tile types" whose sides are labeled with glue "labels" and "strengths." Two tiles placed next to each other *interact* if the glue labels on their abutting sides match, and a tile *binds* to an assembly if the total strength on all of its interacting sides exceeds the ambient "temperature," equal to 2 in this paper. See Section 2 for a formal definition.

Winfree [21] demonstrated the computational universality of the aTAM by showing how to simulate an arbitrary cellular automaton with a tile assembly system. Building on these connections to computability, Rothemund and Winfree [12] investigated the minimum number of tile types needed to uniquely assemble an $n \times n$ square. Utilizing the theory of Kolmogorov complexity, they show that for any algorithmically random n, $\Omega\left(\frac{\log n}{\log \log n}\right)$ tile types are required to uniquely assemble an $n \times n$ square, and Adleman, Cheng, Goel, and Huang [1] exhibit a construction showing that this lower bound is asymptotically tight.

Real-life implementations of the aTAM involve (at the present time) creating tile types out of DNA double-crossover molecules [13], copies of which can be created at an exponential rate using the polymerase chain reaction (PCR) [14]. PCR technology has advanced to the point where it is automated by machines, meaning that *copies* of tiles are easy to supply, whereas the number of distinct tile *types* is a precious resource, costing much more lab time to create. Therefore, effort has been put towards developing methods of "programming" tile sets through methods other than hard-coding the desired behavior into the tile types. Such methods include *temperature programming* [6], [18], which involves changing the ambient temperature through the assembly process in order to alter which bonds are possible to break or create, and *staged assembly* [5], which involves preparing different assemblies in different test tubes, which are then mixed after reaching a terminal state. Each of these models allows a *single* tile set to be reused for assembling different structures by programming it with different environmental conditions affecting the behavior of the tiles.

The model used in this paper is known as *tile concentration programming*. If the tile assembly system is nondeterministic – if intermediate assemblies exist in which more than one tile type is capable of binding to the same position – and if the solution is well-mixed, then the relative concentrations of these tile types determine the probability that each tile type will be the one to bind.

Tile concentrations affect the expected time before an assembly is completed (such a model is considered in [1] and [2], for instance), but we ignore such running time considerations in the present paper. We instead focus on using the biased randomness of tile concentrations to guide a probabilistic shape-building algorithm, subject a certain kind of "geometric *space* bound"; namely, that the algorithm must be executed within the confines of the shape being assembled. This restriction follows from the monotone nature of the aTAM: once a tile attaches to an assembly, it never

detaches. Chandran, Gopalkrishnan, and Reif [4] show that a one-dimensional line of expected length n can be assembled using $\Theta(\log n)$ tile types, subject to the restriction that all tile concentrations are equal. Furthermore, they show that this bound is tight for *all* n. Becker, Rapaport, and Rémila [2] show that there is a *single* tile assembly system $\mathcal{T}$ such that, for all $n \in \mathbb{Z}^+$, setting the tile concentrations appropriately causes $\mathcal{T}$ to assemble an n' x n' square, such that n' has expected value n. However, n' will have a large deviation from n with non-negligible probability. Kao and Schweller [7] improve this result by constructing, for each $\delta, \epsilon > 0$, a tile assembly system $\mathcal{T}$ such that setting the tile concentrations appropriately causes $\mathcal{T}$ to assemble an n' x n' square, where $(1-\epsilon)n \leq n' \leq (1+\epsilon)n$ with probability at least $1-\delta$, for sufficiently large $n \in \mathbb{Z}^+$.

Kao and Schweller asked whether a constant-sized tile assembly system could be constructed that, through tile concentration programming, would assemble a square of dimensions *exactly* $n \times n$, with high probability. We answer this question affirmatively, showing that, for each $\delta > 0$, there is a tile assembly system $\mathcal{T}$ such that, for sufficiently large $n \in \mathbb{Z}^+$, there is an assignment of tile concentrations to $\mathcal{T}$ such that $\mathcal{T}$ assembles an $n \times n$ square with probability at least $1-\delta$.

Kao and Schweller also asked whether arbitrary finite connected shapes, possibly scaled by factor $c \in \mathbb{N}$ (depending on the shape) by replacing each point in the shape with a $c \times c$ block of points, could be assembled from a constant tile set through concentration programming. Our construction answering the first question computes the binary expansion of n with high probability in a self-assembled rectangle of height $O(\log n)$ and width $O(n^{2/3})$. By assembling this structure within the "seed block" of the construction of [17], our construction can easily be combined with that of [17] to answer this question affirmatively as well, by replacing the number n with a program that outputs a list of points in the shape, and using this as the "seed block" of the construction of [17]. We omit a detailed construction in this extended abstract.

This paper is organized as follows. Section 2 provides background definitions and notation for the aTAM and tile concentration programming. Section 3 provides the construction and proof of correctness. Section 4 concludes the paper, discusses practical limitations of the construction and potential improvements, and suggests non-square structures that can be assembled with the same technique.

2. The Tile Assembly Model and Tile Concentration Programming

We give a brief sketch of the Tile Assembly Model that is adequate for reading this paper. More details and discussion may be found in [8], [11], [12], [21]. Our notation is that of [8], which provides a self-contained introduction to the Tile Assembly Model for the reader unfamiliar with the model.

All logarithms in this paper are base 2. We work in the 2-dimensional discrete space $\mathbb{Z}^2$. Define the set $U_2 = \{(0,1),(1,0),(0,-1),(-1,0)\}$ to be the set of all *unit vectors*, i.e., vectors of length 1 in $\mathbb{Z}^2$. We write $[X]^2$ for the set of all 2-element subsets of a set X. All *graphs* in this paper are undirected graphs, i.e., ordered pairs $G = (V, E)$, where V is the set of *vertices* and $E \subseteq [V]^2$ is the set of *edges*.

Intuitively, a tile type t is a unit square that can be translated, but not rotated, having a well-defined "side $\vec{u}$" for each $\vec{u} \in U_2$. Each side $\vec{u}$ of t has a "glue" with "label" $\text{label}_t(\vec{u})$ – a string over some fixed alphabet Σ – and "strength" $\text{str}_t(\vec{u})$ – a non-negative integer – specified by its type t. Two tiles t and t' that are placed at the points $\vec{a}$ and $\vec{a}+\vec{u}$ respectively, *bind* with *strength* $\text{str}_t(\vec{u})$ if and only if $(\text{label}_t(\vec{u}), \text{str}_t(\vec{u})) = (\text{label}_{t'}(-\vec{u}), \text{str}_{t'}(-\vec{u}))$. In our figures, we follow Winfree's convention of representing strength-0 bonds with dashed lines, strength-1 bonds with single lines, and strength-2 bonds with double lines.

Given a set T of tile types, an *assembly* is a partial function $\alpha : \mathbb{Z}^2 \dashrightarrow T$, with points $\vec{x} \in \mathbb{Z}^2$ at which $\alpha(\vec{x})$ is undefined interpreted to be empty space, so that $\text{dom } \alpha$ is the set of points with tiles. α is *finite* if $|\text{dom } \alpha|$ is finite. For assemblies α and α', we say that α is a *subassembly* of α', and write $\alpha \sqsubseteq \alpha'$, if $\text{dom } \alpha \subseteq \text{dom } \alpha'$ and $\alpha(\vec{x}) = \alpha'(\vec{x})$ for all $x \in \text{dom } \alpha$. α' is a *single-tile extension* of α if $\alpha \sqsubseteq \alpha'$ and $\text{dom } \alpha' - \text{dom } \alpha$ is a singleton set. In this case, we write $\alpha' = \alpha + (\vec{m} \mapsto t)$, where $\{\vec{m}\} = \text{dom } \alpha' - \text{dom } \alpha$ and $t = \alpha'(\vec{m})$

A *grid graph* is a graph $G = (V, E)$ in which $V \subseteq \mathbb{Z}^2$ and every edge $\{\vec{a}, \vec{b}\} \in E$ has the property that $\vec{a} - \vec{b} \in U_2$. The *binding graph of* an assembly α is the grid graph $G_\alpha = (V, E)$, where $V = \text{dom } \alpha$, and $\{\vec{m}, \vec{n}\} \in E$ if and only if (1) $\vec{m} - \vec{n} \in U_2$, (2) $\text{label}_{\alpha(\vec{m})}(\vec{n} - \vec{m}) = \text{label}_{\alpha(\vec{n})}(\vec{m} - \vec{n})$, and (3) $\text{str}_{\alpha(\vec{m})}(\vec{n} - \vec{m}) > 0$. An assembly is *$\tau$-stable*, where $\tau \in \mathbb{N}$, if it cannot be broken up into smaller assemblies without breaking bonds of total strength at least τ; i.e., if every cut of G_α has weight at least τ, where the weight of an edge is the strength of the sides of tiles

that it connects. In contrast to the model of Wang tiling, the nonnegativity of the strength function implies that glue mismatches between adjacent tiles do not prevent a tile from binding to an assembly, so long as sufficient binding strength is received from the sides of the tile at which the glues match.

Self-assembly begins with a *seed assembly* σ (typically assumed to be finite and τ-stable) and proceeds asynchronously and nondeterministically,[1] with tiles adsorbing one at a time to the existing assembly in any manner that preserves stability at all times, formally modeled as follows.

A *tile assembly system* (*TAS*) is an ordered triple $\mathcal{T} = (T, \sigma, \tau)$, where T is a finite set of tile types, $\sigma : \mathbb{Z}^2 \dashrightarrow T$ is the *seed assembly*, satisfying $|\mathrm{dom}\ \sigma| < \infty$, and $\tau \in \mathbb{N}$ is the *temperature*, equal to 2 in this paper.[2] A *(finite) assembly sequence*[3] in a TAS $\mathcal{T} = (T, \sigma, 2)$ is a finite sequence $\vec{\alpha} = (\alpha_i \mid 1 \leq i \leq k)$ of assemblies in which $\alpha_1 = \sigma$, each α_{i+1} is a single-tile extension of α_i, and each α_i is τ-stable. The *result* of an assembly sequence is $\mathrm{res}(\vec{\alpha}) = \alpha_k$. We write $\mathcal{A}[\mathcal{T}]$ to denote the set of all results of assembly sequences of $\mathcal{T}$, known as the *producible assemblies* of $\mathcal{T}$. An assembly α is *terminal*, and we write $\alpha \in \mathcal{A}_\square[\mathcal{T}]$, if no tile can be stably added to it.

Let $\mathcal{T} = (T, \sigma, \tau)$ be a TAS. A *tile concentration assignment* on $\mathcal{T}$ is a function $\rho : T \to [0, \infty)$.[4] If $\rho(t)$ is not specified explicitly for some $t \in T$, then $\rho(t) = 1$. If $\alpha : \mathbb{Z}^2 \dashrightarrow T$ is a τ-stable assembly such that $t_1, \ldots, t_j \in T$ are the tiles capable of binding to the same position $\vec{m}$ of α,[5] then for $1 \leq i \leq j$, t_i binds at position $\vec{m}$ with probability $\frac{\rho(t_i)}{\rho(t_1)+\ldots+\rho(t_j)}$.[6] ρ induces a probability measure on $\mathcal{A}_\square[\mathcal{T}]$ in the obvious way.[7] For $p \in [0, 1]$ and $X \subseteq \mathbb{Z}^2$, we say X *strictly self-assembles in* $\mathcal{T}(\rho)$ *with probability at least* p if $\Pr_{\alpha \in \mathcal{A}_\square[\mathcal{T}]}[\mathrm{dom}\ \alpha = X] \geq p$. That is, $\mathcal{T}$ self-assembles into a shape equal to X with probability at least p. Note that different two assemblies may have the same shape though they might assign different tile types to the same position.

3. Construction of the Tile Set

This section is devoted to proving the following theorem, which is the main result of this paper.

For all $\delta > 0$ and $n \in \mathbb{Z}^+$, define $r_\delta = \left\lceil \frac{\log \frac{\delta}{8}}{\log 0.9421} \right\rceil$, $c_\delta = 2 + \left\lceil \log \frac{\log \frac{\delta}{8}}{\log 0.717} \right\rceil$, and $k_n = \left\lceil \frac{\lfloor \log n \rfloor + 1}{3} \right\rceil$, and define $b(n, \delta) = \max\left\{r_\delta, 2^{2k_n + c_\delta}\right\} + c_\delta + 3k_n$.

Theorem 3.1. *For all $\delta > 0$, there is a tile assembly system $\mathcal{T}_\delta = (T, \sigma, 2)$ such that, for all integers $n \geq b(n, \delta)$, there is a tile concentration assignment $\rho_n : T \to [0, \infty)$ such that a translation of the set $\{\ (x, y) \mid x, y \in \{1, \ldots, n\}\ \}$ strictly self-assembles in $\mathcal{T}_\delta(\rho_n)$ with probability at least $1 - \delta$.*

Note that for any fixed $\delta > 0$, $b(n, \delta) = O(n^{2/3})$ (the constant in the $O()$ depending on δ), whence $n \geq b(n, \delta)$ for all sufficiently large n.

3.1. Intuitive Idea of the Construction

Kao and Schweller introduced a basic primitive in [7] (refining a lower-precision technique described in [2]), called a *sampling line*. The sampling line allows tile concentrations to encode a natural number whose binary representation can be probably approximately reproduced. Kao and Schweller utilize the sampling line

[1] There are multiple senses in which a tile system can be nondeterministic. The trivial sense is that the location of attachment, if there is more than one candidate, is selected nondeterministically. Such systems may still be deterministic in a stronger sense that they will lead to a unique final assembly. We employ a stronger version of nondeterminism in which the tile capable of binding to a *single* position of an assembly is not fixed; the randomized algorithm we implement relies on this choice being made according to the tile concentrations.

[2] A tile set can be "programmed" with different inputs through selection of an appropriate seed assembly. In this paper, we wish to model the situation in which, once work has been done once to create a single tile set, the tile set can be programmed *entirely* through adjustment of tile concentrations. Hence, our result is stated in terms of the existence of a tile assembly *system*, with a fixed seed assembly (in fact, a single seed tile), that can be used to construct squares of any size, solely by adjusting the tile concentrations.

[3] [8] gives a treatment of the model that allows for infinite assembly sequences, and indeed our construction may result in an infinite assembly sequence, though with probability 0. We simplify the presentation by considering only finite assembly sequences.

[4] Note in particular that we do not require ρ to be a probability measure on T. ρ induces a probability measure as described later on subsets of tiles in T that contend nondeterministically to bind, but there may be more than one such subset, and the relative concentration of a tile from one subset to that of another is irrelevant, since they do not compete.

[5] More precisely, if $\alpha + (\vec{m} \mapsto t_i)$ is τ-stable for some $\vec{m} \notin \mathrm{dom}\ \alpha$ and all $i \in \{1, \ldots, j\}$, but $\alpha + (\vec{m} \mapsto t)$ is not τ-stable for any $t \in T - \{t_1, \ldots, t_j\}$.

[6] This quantity is the *conditional* probability that t_i attaches to position $\vec{m}$, *given* that one of $t_1, \ldots, t_j$ will bind to position $\vec{m}$ at the current stage of self-assembly. We do not use probabilities to model the choice of which position $\vec{m}$ receives a tile; any fair assembly sequence (see [8] for a definition) will suffice.

[7] Formally, let $\alpha \in \mathcal{A}_\square[\mathcal{T}]$ be a producible terminal assembly. Let $A(\alpha)$ be the set of all assembly sequences $\vec{\alpha} = (\alpha_i \mid 1 \leq i \leq k)$ such that $\mathrm{res}(\vec{\alpha}) = \alpha$, with $p_{\vec{\alpha},i}$ denoting the probability of attachment of the tile added to α_{i-1} to produce α_i (noting that $p_{\vec{\alpha},i} = 1$ if the i^{th} tile attached without contention). Then $\Pr[\alpha] = \sum_{\vec{\alpha} \in A(\alpha)} \prod_{i=2}^{k} \frac{1}{|\partial \alpha_i|} p_{\vec{\alpha},i}$. Although this definition assigns each frontier location to be equally probable to receive a tile (that is the source of the term $\frac{1}{|\partial \alpha_i|}$), in the constructions of this paper, any fair assembly sequence will work, even one that biases the choice of frontier location attachment away from uniform.

to encode $n \in \mathbb{N}$ by an approximation $n' \in \mathbb{N}$ such that $(1-\epsilon)n \leq n' \leq (1+\epsilon)n$ with probability at least $1-\delta$.

The idea of our construction is as follows. We will "approximate" only numbers m small enough that the sampling line approximation has sufficient space to be an *exact* computation of m with high probability. The construction of Kao and Schweller can be thought of as estimating n by, in a sense, probabilistically counting to n using independent Bernoulli trials with appropriately fixed success probability; i.e., the probabilities are used to estimate an approximate *unary* encoding of n, which is converted to binary by a counter. Representing n in unary, of course, takes space n, and recovering it probabilistically from tiles subject to randomization requires using much more than space n to overcome the error introduced by randomization. Kao and Schweller use an ingenious technique to spread this estimation out into the center of the $n \times n$ square being built, affording $O(n^2)$ space to approximate n closely. However, that construction lacks the space to compute n exactly, which requires much more than n^2 Bernoulli trials – applying the standard Chernoff bound to the Kao-Schweller sampling line achieves an upper bound of $O(n^5)$ trials – to achieve a sufficiently small estimation error. Hence, attempting to use a sampling line directly to compute n would result in a line containing many more tiles than the n^2 tiles that compose an $n \times n$ square, and no amount of twisting the line will cause it to fit inside the boundaries of the square.

We split n's binary expansion $b(n) = b_1 b_2 \ldots b_{\lfloor \log n \rfloor + 1} \in \{0,1\}^*$ into three subsequences $b_1 b_4 b_7 \ldots$, $b_2 b_5 b_8 \ldots$, and $b_3 b_6 b_9 \ldots$, each of length about $\frac{1}{3} \log n$, and interpret these binary strings as natural numbers $m_1, m_2, m_3 \leq n^{1/3}$ to be estimated. The problem of estimating n is reduced to that of estimating these three numbers. At the same time, we introduce a new sampling line technique that can exactly estimate a number m with high probability using only $O(m^2)$ trials.[8] Since $m_1, m_2, m_3 \leq n^{1/3}$, estimating m_1, m_2, and m_3 will require $O(n^{2/3})$ trials, which fits within the width of an $n \times n$ square for sufficiently large n.

Intuitively, the reason that estimating m_1, m_2, and m_3 creates an improvement over estimating n directly is that the space needed for the unary encodings of numbers whose *binary* length is one-third that of n's does not scale linearly with that length; the *unary* encoding of these numbers scales with $n^{1/3}$, not $n/3$, whence a quadratic increase in the space needed for probabilistic recovery remains sufficiently small ($O(n^{2/3})$) that three such decodings easily fit into space n.

3.2. Probabilistic Decoding of a Natural Number using a Sampling Line

In this section, we describe how to exactly compute a positive integer m probabilistically from tile concentrations that are appropriately programmed to represent m. In our final construction, the sampling line will estimate not one but three integers m_1, m_2, and m_3, as described in Section 3.1, by embedding additional bits into the tiles. However, for the sake of clarity, in this section, we describe how to estimate a single positive integer m, and then describe in Section 3.2.2 how to modify the construction and set the probabilities to allow three numbers to be estimated simultaneously on a single sampling line.

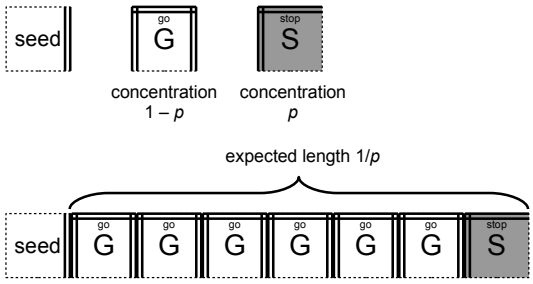

Figure 1. The portion of the basic Kao-Schweller sampling line that controls its length. Two tiles compete nondeterministically to bind to the right of the line, one of which stops the growth, while the other continues, giving the length of the line a geometric distribution.

The basic length-controlling portion of the Kao-Schweller sampling line is shown in Figure 1.[9] A horizontal row of tiles forms to the right of the seed. Two tiles, G ("go") and S ("stop") nondeterministically connect to the right end of the line; G continues the growth, while S stops the growth. If S has concentration

[8] As opposed to the $O(m^5)$ trials that would be required by the Kao-Schweller sampling line. It is possible to use Kao and Schweller's original sampling line to estimate seven numbers – $\lfloor \log n \rfloor + 1$ (the length of the binary expansion of n), and the six numbers m_1 - m_6 encoded by length-$\left\lceil \frac{\lfloor \log n \rfloor + 1}{6} \right\rceil$ substrings of n's binary expansion, each small enough that $m_i^5 = o(n)$ – and to use these numbers to reconstruct n and from that, build an $n \times n$ square. A straightforward and tedious analysis of the constants involved reveals that such a technique can be used to construct $n \times n$ squares for $n \geq 10^{18}$. We achieve much more feasible bounds on n ($\approx 10^7$ for $\delta = 0.01$) using the techniques introduced in this paper, and indeed, better bounds than those required by Kao and Schweller to approximate n, whose construction achieves, for instance, a $(0.01, 0.01)$-approximation only for $n \geq 10^{13}$, according to their analysis.

[9] Our description of the Kao-Schweller sampling line is incomplete, as discussed in the next paragraph.

$p \in [0,1]$ and G has concentration $1-p$, then the length L of the line is a geometric random variable with expected value $1/p$. By setting p appropriately, $\mathrm{E}[L]$ can be controlled, but not precisely, since a geometric random variable may have a deviation from the expected value that is too large for our purposes.

Kao and Schweller allow a third tile type to bind within the sampling line, which does the actual sampling for computing a natural number, but our construction splits this sampling into a separate set of tiles that forms above the line. The sampling portion is discussed in Section 3.2.2. For the present time, we restrict our discussion to controlling the length of the line.

3.2.1. More Precisely Controlling the Sampling Line Length: Our goal is to control L, the length of the sampling line, such that, by setting tile concentrations appropriately, we may ensure that L lies between 2^{a-1} and 2^a with high probability, for an $a \in \mathbb{Z}^+$ of our choosing (which will be influenced by the number n we are estimating). That is, we may ensure that the number of bits required to represent L is computed precisely, even if the exact value of L varies widely within the interval $[2^{a-1}, 2^a)$. We then attach a counter – a group of tiles that measures the length of the line by counting in binary – to the north of the line that measures L until the final stopping tile. The stop signal is not intended to stop the counter immediately, but rather to signal that the counter should continue until it reaches the next power of 2 – i.e., the next time a new most significant bit is required – and then stop. Hence, we may choose an arbitrary power of 2 and set tile concentrations to ensure that the counter counts to that value and then stops.

To increase the precision with which we control L, we use not one but many stages of "go" and "stop" tiles, $G_1, S_1, G_2, S_2, \ldots, G_r, S_r$. The construction is shown in Figure 2. G_i and S_i each compete to bind to the right of S_{i-1} and G_i. S_i signals a transition to the next stage $i+1$, with S_r stopping the growth of the line after r stages. Therefore, the sequence of tiles to the right of the seed is a string described by the regular expression $G_1^* S_1 G_2^* S_2 \ldots G_r^* S_r$. Each S_i has concentration p, and the remaining G_i tiles each have concentration $1-p$. The length L of the line is a *negative binomial* random variable[10] with parameters r, p (see [9]) with expected value r/p by linearity of expectation; i.e., its length is the number of Bernoulli trials required before exactly r successes, provided each Bernoulli trial has success probability p.

Let $N, R \in \mathbb{N}$ and $p \in [0,1]$. A binomial random variable $\mathcal{B}(N,p)$ (the number of successes after N Bernoulli trials, each having success probability p) is related to a negative binomial random variable $\mathcal{N}(R,p)$ (the number of trials before exactly R successes) by the relationships

$$\Pr[\mathcal{N}(R,p) < N] = \Pr[\mathcal{B}(N,p) > R], \quad (3.1)$$
$$\Pr[\mathcal{N}(R,p) > N] = \Pr[\mathcal{B}(N,p) < R]. \quad (3.2)$$

Thus, Chernoff bounds that provide tail bounds for binomial distributions can be applied to negative binomial distributions via (3.1) and (3.2).

To cause L to fall in the interval $[2^{a-1}, 2^a)$, we must set its expected length $\overline{L}$ (by setting $p = r/\overline{L}$) to be such that the r^{th} success occurs when the line has length in the interval $[2^{a-1}, 2^a)$. Note that pN is the expected number of successes in the first N tiles of the line; i.e., it is the expected number of successes in exactly N Bernoulli trials.

We define ϵ and ϵ' so that $\overline{L} = (1+\epsilon)2^{a-1} = (1-\epsilon')2^a$ and the two error probabilities derived below are approximately equal; $\epsilon \approx 0.442695$ and $\epsilon' \approx 0.2786525$ suffice. The event that $L < 2^{a-1}$ is equivalent to the event that 2^{a-1} Bernoulli trials are conducted (with expected number of successes $p2^{a-1}$) with at least r successes. By (3.1) and the Chernoff bound [9, Theorem 4.4, part 1],

$$\begin{aligned}
\Pr\left[L < 2^{a-1}\right] &= \Pr\left[r > (1+\epsilon)p2^{a-1}\right] \\
&\leq \left(\frac{e^\epsilon}{(1+\epsilon)^{1+\epsilon}}\right)^{p2^{a-1}} \\
&= \left(\frac{e^\epsilon}{(1+\epsilon)^{1+\epsilon}}\right)^{r2^{a-1}/\overline{L}} \\
&= \left(\frac{e^\epsilon}{(1+\epsilon)^{1+\epsilon}}\right)^{r2^{a-1}/\left((1+\epsilon)2^{a-1}\right)} \\
&= \left(\frac{e^\epsilon}{(1+\epsilon)^{1+\epsilon}}\right)^{r/(1+\epsilon)} \\
&< 0.9421^r.
\end{aligned}$$

The event that $L \geq 2^a$ is equivalent to the event that 2^a Bernoulli trials are conducted (with expected number of successes $p2^a$) with fewer than r successes. To bound the probability that L is too large, we use (3.2) and the Chernoff bound for deviations below the

[10] The term *negative* is misleading; a negative binomial random variable is better described (informally) as the *inverse* of a binomial random variable, if one thinks of a binomial random variable as being like a function that maps a number of Bernoulli trials to a number of successes. A negative binomial random variable maps a number of successes to the number of trials necessary to achieve that number of successes.

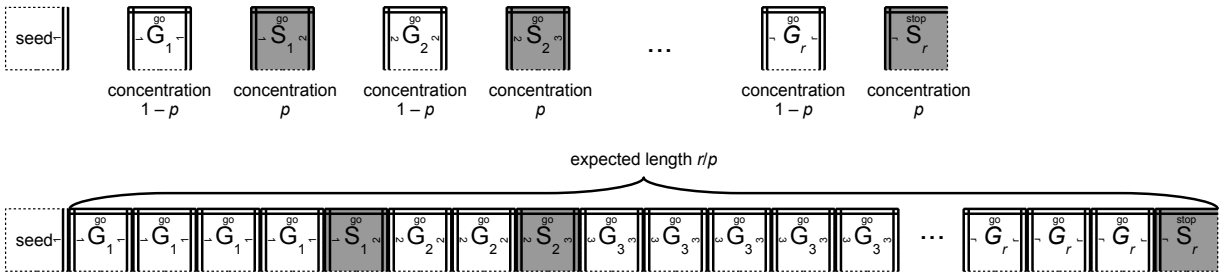

Figure 2. The portion of the sampling line of our construction that controls its length. r stages each have expected length $1/p$, making the expected total length r/p, but more tightly concentrated about that expected length than in the case of one stage.

mean [9, Theorem 4.5, part 1],

$$\begin{aligned}
\Pr[L \geq 2^a] &= \Pr[r \leq (1-\epsilon')p2^a] \\
&\leq \left(\frac{e^{-\epsilon'}}{(1-\epsilon')^{1-\epsilon'}}\right)^{p2^a} \\
&= \left(\frac{e^{-\epsilon'}}{(1-\epsilon')^{1-\epsilon'}}\right)^{r2^a/\overline{L}} \\
&= \left(\frac{e^{-\epsilon'}}{(1-\epsilon')^{1-\epsilon'}}\right)^{r2^a/\left((1+\epsilon)2^{a-1}\right)} \\
&= \left(\frac{e^{-\epsilon'}}{(1-\epsilon')^{1-\epsilon'}}\right)^{2r/(1+\epsilon)} \\
&< 0.9421^r.
\end{aligned}$$

By the union bound,

$$\Pr[L \notin [2^{a-1}, 2^a)] < 2 \cdot 0.9421^r \tag{3.3}$$

Therefore, by setting r sufficiently large, we can exponentially decrease the probability that L falls outside the range $[2^{a-1}, 2^a)$, independently of a. For example, letting $r = 113$ leads to $\Pr\left[L \notin [2^{a-1}, 2^a)\right] < 0.0025$. Since r is a constant depending only on δ, it can be encoded into the tile types as shown in Figure 2.

3.2.2. Computing a Number Exactly using a Sampling Line: As stated previously, our goal is that, with a sampling line of length $O(m^2)$, we can exactly compute a number m. The idea is shown in Figure 3, and is inspired by the sampling line of Kao and Schweller [7] but can estimate a number more precisely using a given length, as well as having a length that is itself controlled more precisely by the technique of Section 3.2.1. The length-controlling portion of the sampling line of length L will control a counter placed above the sampling line, which counts to the next power of 2 greater than L, 2^a. This counter will eventually end up with a total bits before stopping. Let k be the maximum number of bits needed to represent m (k will be about $\frac{1}{3}\log n$ in our application), and let $l = a - k$. We form a row above the row described in Section 3.2.1, which does the sampling. To implement the Bernoulli trials that estimate m, one of two tiles A (the gray tile in Figure 3) or B (the white tile in Figure 3) nondeterministically binds to every position of this row. Set the concentration of A to be $\frac{m2^l+2^{l-1}}{2^a}$ and the concentration of B to be $1 - \frac{m2^l+2^{l-1}}{2^a}$. We embed a second counter – the sampling counter – within the primary counter. Whenever A appears, the sampling counter increments, and when B appears it does not change. Let M be the random variable representing the final value of the sampling counter. Then M is a binomial random variable with $\mathrm{E}[M] = m2^l + 2^{l-1}$.

We will choose k and l so that the most significant k bits of the sampling counter will almost certainly represent m. Intuitively, the least significant l bits of M "absorb" the error. This will occur if $m2^l \leq M < (m+1)2^l$. Note that $m < 2^k$. Let $\varepsilon = \frac{1}{2m}$. Then the Chernoff bound [9, Theorems 4.4/4.5, part 2] and the union bound tell us that

$$\begin{aligned}
&\Pr\left[M \geq (m+1)2^l \text{ or } M < m2^l\right] \\
&= \Pr[M \geq (1+\varepsilon)\mathrm{E}[M] \text{ or } M < (1-\varepsilon)\mathrm{E}[M]] \\
&\leq e^{-\mathrm{E}[M]\varepsilon^2/3} + e^{-\mathrm{E}[M]\varepsilon^2/2} \\
&< e^{-m2^l\left(\frac{1}{2m}\right)^2/3} + e^{-m2^l\left(\frac{1}{2m}\right)^2/2} \\
&= e^{-\frac{2^{l-2}}{3m}} + e^{-\frac{2^{l-2}}{2m}} \\
&< e^{-2^{l-k-2}/3} + e^{-2^{l-k-2}/2}
\end{aligned}$$

Let $c \in \mathbb{N}$ be a constant. By setting $l = k + c$, the probability of error decreases exponentially in c:

$$\begin{aligned}
&\Pr\left[M \geq (m+1)2^l \text{ or } M < m2^l\right] \\
&< e^{-2^{c-2}/3} + e^{-2^{c-2}/2} < 2 \cdot 0.717^{2^{c-2}}. \qquad (3.4)
\end{aligned}$$

For instance, letting $c = 6$ bounds the left-hand side of (3.4) below 0.0052.

The number of samples is $2^a = 2^{2k+c} = O((2^k)^2)$. Since $m < 2^k$, integers m such that $m^2 \ll n$ can be

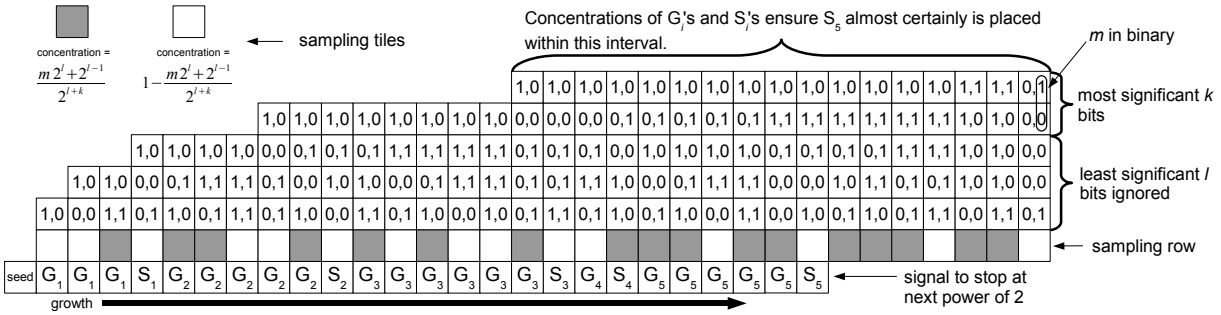

Figure 3. Computing the natural number $m = 2$ from tile concentrations using a sampling line. For brevity, glue strengths and labels are not shown. Each column increments the primary counter, represented by the bits on the left of each tile, and each gray tile increments the sampling counter, represented by the bits on the right of each tile. The number of bits at the end is $l + k$, where c is a constant coded into the tile set, and k depends on m, and $l = k + c$. The most significant k bits of the sampling counter encode m. In this example, $k = 2$ and $c = 1$.

"probably exactly computed" using much fewer than n Bernoulli trials, and can therefore be computed by a sampling line without exceeding the boundaries of an $n \times n$ square.

3.3. Computing n Exactly

We have shown how to compute a number m exactly using a sampling line of length $O(m^2)$ and width $O(\log m)$. To compute n, the dimensions of the square, we must compute m_1, m_2, and m_3, which are the numbers represented by the bits of the binary expansion of n at positions congruent to 1 mod 3, 2 mod 3, and 0 mod 3, respectively. To compute all three of these numbers, we embed two extra sampling counters into the double counter, in addition to the sampling counter described in Section 3.2, to create a quadruple counter. This requires 8 sampling tiles instead of 2, in order to represent each of the possible outcomes of conducting three simultaneous Bernoulli trials, each trial used for estimating one of m_1, m_2, or m_3.

Given $i \in \{1, 2, 3\}$, let $b_i \in \{0, 1\}$ denote the outcome of the i^{th} of three simultaneous Bernoulli trials, and let $p_i(b_i)$ denote the probability we would like to associate with that outcome. As noted in Section 3.2.2, the values of the c_i's are given by $p_i(1) = \frac{m_i 2^l + 2^{l-1}}{2^a}$, and $p_i(0) = 1 - p_i(1)$.

Since each of the three simultaneous Bernoulli trials is independent, we can calculate the appropriate concentration of the tile representing the three outcomes by multiplying the three outcome probabilities together. Then the required concentration of the tile representing outcomes b_1, b_2, b_3 is given by $p_1(b_1) \cdot p_2(b_2) \cdot p_3(b_3)$.

Once the values m_1, m_2, and m_3 are computed, we must remove the c least significant (bottom) bits from the bottom of the primary counter. Since c is a constant depending only on δ, it can be encoded into the tile types. We must then remove the bottom half of the remaining bits.[11] At this point, the concatenation of the bits on the tiles represent the binary expansion of n. Rather than expand them out to use three times as many tiles, we simply translate each of them to an octal digit, giving the octal representation of n, with one octal digit per tile replacing the three bits per tile. Finally, this representation of n is rotated 90 degrees counter-clockwise, used as the initial value for a decrementing, upwards-growing, base-8 counter, and used to fill in an $n \times n$ square using the standard construction [12]. Rotating n to face up starts the counter $2k+2$ tiles from the bottom of the construction so far. Furthermore, testing whether the counter has counted below 0 requires counting once beyond 0, using 2 more rows than the starting value of the counter. Therefore, to ensure that exactly an $n \times n$ square is formed, the value $n - 2k - 4$, rather than n exactly, is programmed into the tile concentrations to serve as the start value of the upwards-growing counter. An outline of this construction is shown in Figure 4.

It is routine to verify that the choice of the parameters c, k, and r described just before Theorem 3.1 are sufficient to obtain the bound $b(n, \delta)$ of Theorem 3.1.

A simulated implementation of this tile assembly system using the ISU TAS Tile Assembly Simulator [10] is available at http://www.cs.iastate.edu/~lnsa/software.html. The tile set uses approximately $4500+9c+4r$ tile types, where r and c are calculated from δ as above.

[11] Isolating the most significant half of the bits can be done using a tile set similar to the algorithm one might use to program a single-tape Turing machine to compute the function $0^{2n} \mapsto 0^n$.

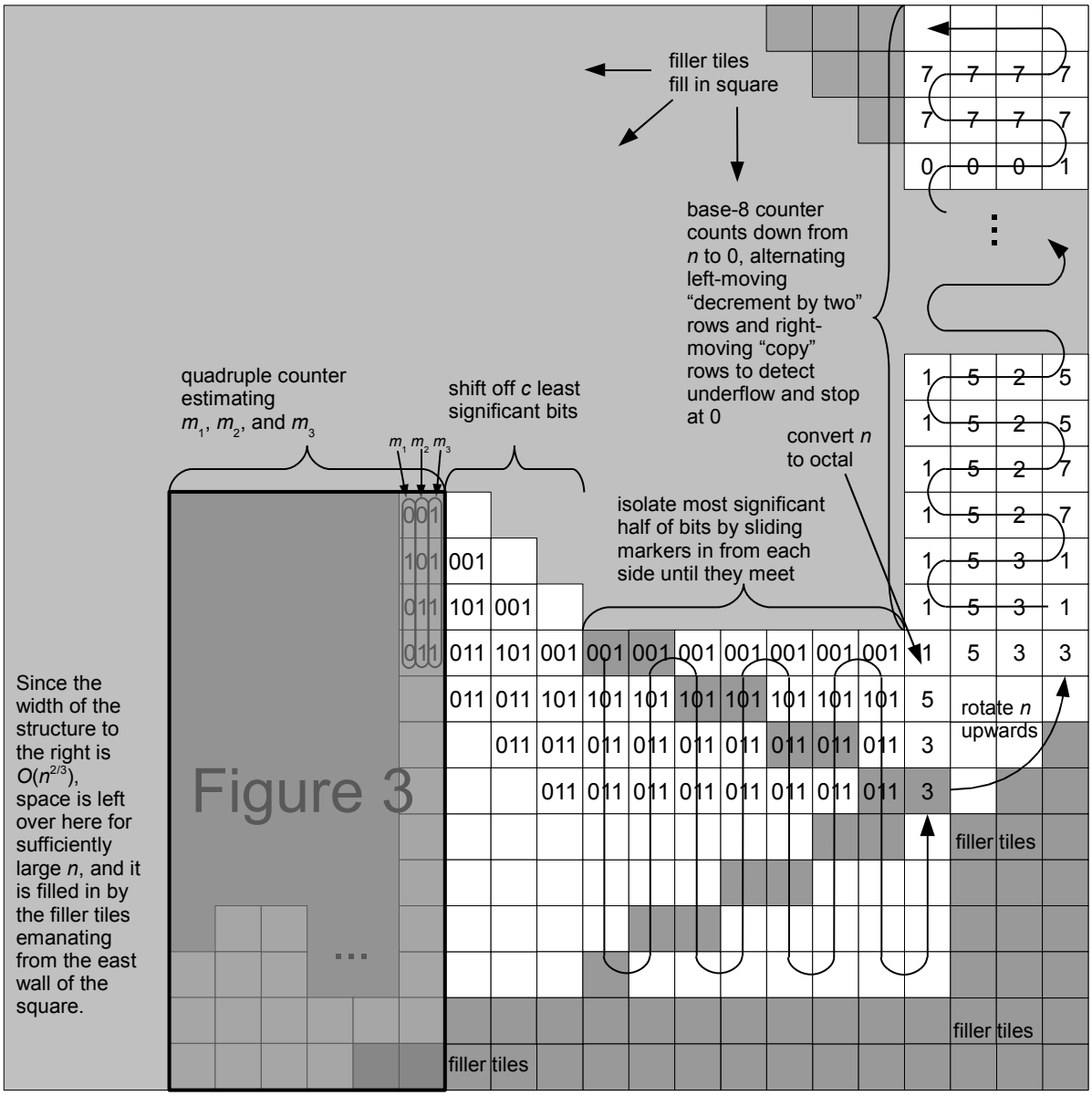

Figure 4. High-level overview of the entire construction, not at all to scale. For brevity, glue strengths and labels are not shown. The double counter number estimator of Figure 3 is embedded with two additional counters to create a quadruple counter estimating m_1, m_2, and m_3, shown as a box labeled as "Figure 3" in the above figure. In this example, $m_1 = 4$, $m_2 = 3$, and $m_3 = 15$, represented vertically in binary in the most significant 4 tiles at the end of the quadruple counter. Concatenating the bits of the tiles results in the string 001101011011, the binary representation of 859, which equals $n - 2k - 4$ for $n = 871$, so this example builds an 871 x 871 square. (Actually, 871 is too small to work with our construction, so the counter will exceed length 871, but we choose small numbers to illustrate the idea more clearly.) Once the counter ends, c tiles ($c = 3$ in this example) are shifted off the bottom, and the top half of the tiles are isolated ($k = 4$ in this example). Each remaining tile represents three bits of n, which are converted into octal digits, rotated to face upwards, and then used to initialize a base-8 counter that builds the east wall of the square. Filler tiles cover the remaining area of the square.

4. Conclusion

We have described how a single tile set in Winfree's abstract tile assembly model, appropriately "programmed" by setting tile concentrations, exactly assembles an $n \times n$ square with high probability, for any sufficiently large n.

The focus of the present paper is on conceptual clarity. We have therefore described the simplest (i.e., easiest to understand, but not necessarily smallest) version of the tile assembly system that achieves the desired *asymptotic* result that an $n \times n$ square assembles with high probability for sufficiently large n. We now observe that this theoretical result could be improved in practice by complicating the tile set.

Our implementation of the tile set uses approximately $4500 + 9c + 4r$ tile types, where, for example, $r = 113$ and $c = 7$ are sufficient to achieve error probability $\delta \leq 0.01$. The tiles are so numerous because of the need to simultaneously represent 4 bits in a tile, in addition to information such as the significance of the bit (MSB, LSB, or interior bit), and doing computation such as addition, which requires tiles that can handle the 2^8 possible input bit + carry signals. Putting together a few such modules of tile sets results in thousands of tiles before too long. The number of tile types could be reduced by splitting the estimation of m_1, m_2, and m_3 into three distinct geometrical regions, so that each tile is required to remember less information. This would complicate the tile set, as it would require more shifting tricks to ensure sufficient room for all counters, and would require bringing the bits back together again at the end, but it would likely reduce the number of tile types.

A large value of n is required to achieve a probability of success at least $(1 - \delta)$ for reasonably small δ; $n > 8 \cdot 10^6$ is required to estimate n with 99% chance of correctness. This shortcoming can be compensated in a number of ways.

In a similar spirit to the linear speedup theorem, more than three simultaneous Bernoulli trials may be conducted with each sampling tile. For example, conducting 6 Bernoulli trials with each sampling tile would estimate two bits of $m_1, \ldots, m_3$ with per sampling tile, rather than one bit, halving the required length of the sampling line. This would result in a prohibitively large tile set, however; as the number of tile types increases exponentially with the number of simultaneous Bernoulli trials per tile type.

A conceptually simpler and practically more feasible improvement is to use 0/1-valued tile concentrations to simulate tile *type* programming (i.e., designing tile types specially to build a particular size square, as in [12]) for small values of n, by including tile types that deterministically construct an $n \times n$ square for each small n, setting concentrations of those tiles to be 1 and setting concentrations of all other tiles to be 0. Though this solution lacks the "feel" of tile concentration programming, it is likely that real-life implementations of tile concentration programming will need to use such hard-coding tricks for smaller structures that lack the space to carry out the amount of sampling required to reconstruct precise inputs solely from tile concentrations.

An alternate improvement to the tile set would be to combine the present technique with the Kao-Schweller technique of building a sampling line inside of a square, to more efficiently use the n^2 space available to carry out the estimation. However, square-building is not necessarily the only application of this technique, as discussed next.

The primary novel contribution of this paper is a tile set that, through appropriate tile concentration programming, forms a thin structure of length $O(n^{2/3})$ and height $O(\log n)$,[12] whose rightmost tiles encode the value of n. The number n could be used to assemble useful structures other than squares. For the task of building a square, this construction wastes the $\approx n^2$ space available above the thin rectangle, but for computing other structures, it may be advantageous that the rectangle is kept thin. For instance, biochemists routinely use filters (e.g., Millipore Ultrafiltration Membranes) and porous resins [3] to separate proteins based on size, in order to isolate one particular protein for study. The ability to precisely control the size of the filter holes or resin beads would allow for more targeted filtering of proteins than is possible at the present time. DNA is likely too reactive with amino acids to be used as the substrate for such a structure, so an implementation of the tile assembly model not based on DNA would be required for such a technique.

Similarly, polyacrylamide gel electrophoresis [16], another technique for discriminating biological molecules on the basis of size, requires molecular mass size markers, which are control molecules of known molecular mass, in order to compare against the molecule of interest on the gel. At the present time,

[12] By partitioning n's binary representation into t rather than three subsequences, for $t \in \mathbb{N}$ a constant, the number of trials needed to estimate n is $O(n^{2/t})$. However, the constant factors in the $O()$ increase, making the technique even less feasible for small values of n. But if some application requires an asymptotically very short line, the line can be made length $O(n^{\varepsilon})$ for any $\varepsilon > 0$ using this technique.

some naturally-occurring molecules of known mass are used, but their masses are not controllable, and the ability to quickly and easily assemble molecules of precisely a desired target mass would be useful in experiments requiring mass markers that differ from the standards. Again, DNA is a special case in which this idea is unnecessary, since precise standards have been developed for DNA gels (e.g., Novagen DNA Markers). But the tile assembly model may one day be implemented using substances that are appropriate for a protein gel.

Acknowledgments. I thank Jack Lutz for helpful advice during the preparation of this paper, and Julie Hoy for suggesting potential biochemical applications. I also thank Pavan Aduri and Srikanta Tirthapura for allowing me to audit their classes on randomness and computation, whose subject matter proved very useful in devising the techniques of this paper. This research was supported in part by National Science Foundation Grants 0652569, 0728806, and CCF:0430807.

REFERENCES

[1] L. Adleman, Q. Cheng, A. Goel, and M.-D. Huang, "Running time and program size for self-assembled squares," in *STOC '01: Proceedings of the thirty-third annual ACM Symposium on Theory of Computing*. New York, NY, USA: ACM, 2001, pp. 740–748.

[2] F. Becker, I. Rapaport, and E. Rémila, "Self-assembling classes of shapes with a minimum number of tiles, and in optimal time," in *Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, 2006, pp. 45–56.

[3] D. M. Bollag, "Gel-filtration chromatography," *Methods in Molecular Biology*, vol. 36, pp. 1–9, November 1994. [Online]. Available: http://www.springerprotocols.com/Abstract/doi/10.1385/0-89603-274-4:1

[4] H. Chandran, N. Gopalkrishnan, and J. H. Reif, "The tile complexity of linear assemblies," in *36th International Colloquium on Automata, Languages and Programming*, vol. 5555, 2009.

[5] E. D. Demaine, M. L. Demaine, S. P. Fekete, M. Ishaque, E. Rafalin, R. T. Schweller, and D. L. Souvaine, "Staged self-assembly: nanomanufacture of arbitrary shapes with $O(1)$ glues," *Natural Computing*, vol. 7, no. 3, pp. 347–370, 2008.

[6] M.-Y. Kao and R. T. Schweller, "Reducing tile complexity for self-assembly through temperature programming," in *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2006), Miami, Florida, Jan. 2006, pp. 571-580*, 2007.

[7] ——, "Randomized self-assembly for approximate shapes," in *International Colloqium on Automata, Languages, and Programming (ICALP)*, ser. Lecture Notes in Computer Science, L. Aceto, I. Damgård, L. A. Goldberg, M. M. Halldrsson, A. Inglfsdttir, and I. Walukiewicz, Eds., vol. 5125. Springer, 2008, pp. 370–384. [Online]. Available: http://dblp.uni-trier.de/db/conf/icalp/icalp2008-1.html#KaoS08

[8] J. I. Lathrop, J. H. Lutz, and S. M. Summers, "Strict self-assembly of discrete Sierpinski triangles," *Theoretical Computer Science*, vol. 410, pp. 384–405, 2009.

[9] M. Mitzenmacher and E. Upfal, *Probability and Computing*. Cambridge University Press, 2005.

[10] M. J. Patitz, "Simulation of self-assembly in the abstract tile assembly model with ISU TAS," in *6th Annual Conference on Foundations of Nanoscience: Self-Assembled Architectures and Devices (Snowbird, Utah, USA, April 20-24 2009)*., 2009.

[11] P. W. K. Rothemund, "Theory and experiments in algorithmic self-assembly," Ph.D. dissertation, University of Southern California, December 2001.

[12] P. W. K. Rothemund and E. Winfree, "The program-size complexity of self-assembled squares (extended abstract)," in *STOC '00: Proceedings of the thirty-second annual ACM Symposium on Theory of Computing*. New York, NY, USA: ACM, 2000, pp. 459–468.

[13] P. W. Rothemund, N. Papadakis, and E. Winfree, "Algorithmic self-assembly of DNA Sierpinski triangles," *PLoS Biology*, vol. 2, no. 12, pp. 2041–2053, 2004.

[14] J. Sambrook and D. Russell, *Molecular Cloning: A Laboratory Manual*. Cold Spring Harbor Laboratory Press, 2001.

[15] N. C. Seeman, "Nucleic-acid junctions and lattices," *Journal of Theoretical Biology*, vol. 99, pp. 237–247, 1982.

[16] A. Shapiro, E. Viñuela, and J. M. Jr., "Molecular weight estimation of polypeptide chains by electrophoresis in SDS-polyacrylamide gels," *Biochem Biophys Res Commun.*, vol. 28, pp. 815–820, November 1967.

[17] D. Soloveichik and E. Winfree, "Complexity of self-assembled shapes," *SIAM Journal on Computing*, vol. 36, no. 6, pp. 1544–1569, 2007.

[18] S. M. Summers, "Reducing tile complexity for the self-assembly of scaled shapes through temperature programming," Computing Research Repository, Tech. Rep. 0907.1307, 2009. [Online]. Available: http://arxiv.org/abs/0907.1307

[19] H. Wang, "Proving theorems by pattern recognition – II," *The Bell System Technical Journal*, vol. XL, no. 1, pp. 1–41, 1961.

[20] ——, "Dominoes and the AEA case of the decision problem," in *Proceedings of the Symposium on Mathematical Theory of Automata (New York, 1962)*. Polytechnic Press of Polytechnic Inst. of Brooklyn, Brooklyn, N.Y., 1963, pp. 23–55.

[21] E. Winfree, "Algorithmic self-assembly of DNA," Ph.D. dissertation, California Institute of Technology, June 1998.

APPENDIX

1. Choice of Parameters

We now derive the settings of various parameters required to achieve a desired success probability and derive lower bounds on n necessary to allow the space required by the construction. To ensure probability of failure at most δ, we pick r, the number of stages of stopping tiles that must attach before the primary counter is sent the stop signal, so that $2 \cdot 0.9421^r \leq \frac{\delta}{4}$ as in (3.3):

$$r = \left\lceil \frac{\log \frac{\delta}{8}}{\log 0.9421} \right\rceil.$$

For example, choosing $r = 113$ achieves probability of error $\delta/4$ (in ensuring the counter stops between the numbers 2^{a-1} and 2^a) at most 0.0025.

To ensure that each of m_1, m_2, and m_3 are computed exactly, we set c, the number of extra bits used in the primary counter beyond $2k$, such that $e^{-2^{c-2}/3} + e^{-2^{c-2}/2} \leq \frac{\delta}{4}$, as in (3.4), or more simply, such that $2 \cdot 0.717^{2^{c-2}} \leq \frac{\delta}{4}$; i.e., set

$$c = 2 + \left\lceil \log \frac{\log \frac{\delta}{8}}{\log 0.717} \right\rceil.$$

For example, choosing $c = 7$ achieves probability of error $\delta/4$ (in ensuring that m_1 is computed correctly) at most 0.0025 (in fact, at most 0.000005).

By the union bound, the length of the sampling line and the values of m_1, m_2, and m_3 are computed with sufficient precision to compute the exact value of n with probability at least $1 - \delta$. The example values of r and c given above achieve $\delta \leq 0.01$.

The choices of r and c imply a lower bound on the value of n necessary to allow sufficient space to carry out the construction. Clearly the counter must reach at least value r, since there are r different stopping stages. The more influential factor will be the value c, which doubles the space necessary to run the counter each time it is incremented by 1. n requires $\lfloor \log n \rfloor + 1$ bits to represent, but our estimation will be a string of length the next highest multiple of 3 above $\lfloor \log n \rfloor + 1$. Therefore, each of m_1, m_2, and m_3 requires

$$k = \left\lceil \frac{\lfloor \log n \rfloor + 1}{3} \right\rceil$$

bits to represent. Recall that the primary counter will have height $2k + c$ and count to 2^{2k+c} (so long as $r \leq 2^{2k+c}$). Then, c columns are required to shift off the constant c bits from the least significant bits of the counter, and $2k$ columns are required to shift off the least significant half of the bits of the counter to isolate the k most significant bits. k columns are needed to translate the groups of three bits into octal and to rotate this string to face upwards for the square-building counter.

Hence, the total length required along the bottom of the square to compute n is $\max\{r, 2^{2k+c}\} + c + 3k$. Expanding out the definitions of r, k, and c derived above gives the lower bound $b(n, \delta)$ on n described in Theorem 3.1.

For sufficiently large n and small enough δ, r is much smaller than 2^{2k+c}, so the latter term dominates. For example, to achieve probability of error $\delta \leq 0.01$ requires $n >$ 8,000,000. According to preliminary experimental tests, in practice, a smaller value of c is required than the theoretical bounds we have derived. For example, if the desired error probability is $\delta = 0.01$, setting $c = 7$ satisfies the analysis given above, but in experimental simulation, $c = 3$ appears to suffice for probability of error at most 0.01, and reduces the space requirements by a factor of $2^{7-3} = 16$. In this case, $n = 9000$ can be computed by a construction that will stay within the 9000 x 9000 square.