| | |
|---|---|
| Title: | Randomized self-assembly |
| Name: | David Doty[1] |
| Affil./Addr. | California Institute of Technology, Pasadena, CA, USA. Supported by NSF grants CCF-1219274, CCF-1162589, and 1317694. |
| Keywords: | randomized; tile complexity; linear assembly |
| SumOriWork: | 2006; Becker, Rapaport, Rémila<br>2008; Kao, Schweller<br>2009; Chandran, Gopalkrishnan, Reif<br>2010; Doty |


# Randomized self-assembly


David Doty[1]

California Institute of Technology, Pasadena, CA, USA. Supported by NSF grants CCF-1219274, CCF-1162589, and 1317694.


## Years aud Authors of Summarized Original Work

2006; Becker, Rapaport, Rémila
2008; Kao, Schweller
2009; Chandran, Gopalkrishnan, Reif
2010; Doty

## Keywords


randomized; tile complexity; linear assembly


## Problem Definition

We use the abstract tile assembly model of Winfree [6], which models the aggregation of monomers called *tiles* that attach one at a time to a growing structure, starting from a single *seed* tile, in which bonds ("glues") on the tile are specific (glues only stick to glues of the same type on other tiles) and cooperative (so that multiple weak glues are necessary to attach a tile). The general idea of *randomized* self-assembly is to use the inherent randomness of self-assembly to help the assembly process. If multiple types of tiles are able to bind to a single binding site, then we assume that their relative concentrations determine the probability that each succeeds. With careful design, we can use the same tile set to create different structures, by changing the concentrations to affect what is likely to assemble. Another use of randomness is in reducing the number of different tile types required to assemble a shape.

### Definitions

A *shape* is a finite, connected subset of $\mathbb{Z}^2$. A *tile type* is a unit square with four sides, each side consisting of a *glue label* (finite string) and a nonnegative integer *strength*.

We assume a finite set $T$ of tile types, but an infinite number of copies of each tile type, each copy referred to as a *tile*. An *assembly* is a positioning of tiles on the integer lattice $\mathbb{Z}^2$; i.e., a partial function $\alpha : \mathbb{Z}^2 \dashrightarrow T$. Write $\alpha \sqsubseteq \beta$ to denote that $\alpha$ is a *subassembly* of $\beta$, which means that $\mathrm{dom}\ \alpha \subseteq \mathrm{dom}\ \beta$ and $\alpha(p) = \beta(p)$ for all points $p \in \mathrm{dom}\ \alpha$. In this case, say that $\beta$ is a *superassembly* of $\alpha$. Two adjacent tiles in an assembly *interact* if the glue labels on their abutting sides are equal and have positive strength. Each assembly induces a *binding graph*, a grid graph whose vertices are tiles, with an edge between two tiles if they interact. The assembly is $\tau$*-stable* if every cut of its binding graph has strength at least $\tau$, where the weight of an edge is the strength of the glue it represents (energy $\tau$ is required to separate the assembly). The $\tau$*-frontier* $\partial^\tau \alpha \subset \mathbb{Z}^2 \setminus \mathrm{dom}\ \alpha$ of $\alpha$ (or *frontier* $\partial\alpha$ when $\tau$ is clear from context) is the set of empty locations adjacent to $\alpha$ at which a single tile could bind stably.

A *tile system* is a triple $\mathcal{T} = (T, \sigma, \tau)$, where $T$ is a finite set of tile types, $\sigma : \mathbb{Z}^2 \dashrightarrow T$ is a *seed assembly* consisting of a single tile (i.e., $|\mathrm{dom}\ \sigma| = 1$), and $\tau \in \mathbb{N}$ is the *temperature*. An assembly $\alpha$ is *producible* if either $\alpha = \sigma$ or if $\beta$ is a producible assembly and $\alpha$ can be obtained from $\beta$ by the stable binding of a single tile. In this case write $\beta \to_1 \alpha$ ($\alpha$ is producible from $\beta$ by the attachment of one tile), and write $\beta \to \alpha$ if $\beta \to_1^* \alpha$ ($\alpha$ is producible from $\beta$ by the attachment of zero or more tiles). If $\alpha$ is producible then there is an *assembly sequence* $\boldsymbol{\alpha} = (\alpha_i \mid 1 \leq i \leq k)$ such that $\alpha_1 = \sigma$, $\alpha_k = \alpha$, and for each $i \in \{1, \ldots, k-1\}$, $\alpha_i \to_1 \alpha_{i+1}$. An assembly is *terminal* if no tile can be $\tau$-stably attached to it. Write $\mathcal{A}[\mathcal{T}]$ to denote the set of all producible assemblies of $\mathcal{T}$, and write $\mathcal{A}_\Box[\mathcal{T}]$ to denote the set of all producible, terminal assemblies of $\mathcal{T}$. We also speak of *shapes* assembled by tile assembly systems, by which we mean $\mathrm{dom}\ \alpha$ if $\alpha \in \mathcal{A}_\Box[\mathcal{T}]$, and we consider shapes to be equivalent up to translation.

We now define the semantics of incorporating randomization into self-assembly. Intuitively, there are two sources of nondeterminism in the model as defined: 1) if $|\partial\alpha| > 1$ then there are multiple binding sites, one of which is nondeterministically selected as the next site to receive a tile, and 2) if multiple tile types could bind to a *single* binding site, then one of them is nondeterministically selected. Both concepts are handled by assigning positive real-valued concentrations to each tile type; ref. [3] gives a full definition that accounts for both of these. However, in the results we discuss, only the latter source of nondeterminism will actually affect the probabilities of various terminal assembly being produced; the binding sites themselves can be picked in an arbitrary order without affecting these probabilities. Thus we state here a simpler definition based on this assumption.

A *tile concentration assignment* on $\mathcal{T}$ is a function $\rho : T \to [0, \infty)$. If $\rho(t)$ is not specified explicitly for some $t \in T$, then $\rho(t) = 1$. If $\alpha$ is a $\tau$-stable assembly such that $t_1, \ldots, t_j \in T$ are the tiles capable of binding to the same position $\mathbf{m} \in \partial\alpha$, then for $1 \leq i \leq j$, $t_i$ binds at position $\mathbf{m}$ with probability $\frac{\rho(t_i)}{\rho(t_1)+\ldots+\rho(t_j)}$. $\rho$ induces a probability measure on $\mathcal{A}_\Box[T]$ in a straightforward way. Formally, let $\alpha \in \mathcal{A}_\Box[T]$ be a producible terminal assembly. Let $A(\alpha)$ be the set of all assembly sequences $\boldsymbol{\alpha} = (\alpha_i \mid 1 \leq i \leq k)$ such that $\alpha_k = \alpha$, with $p_{\boldsymbol{\alpha},i}$ denoting the probability of attachment of the tile added to $\alpha_{i-1}$ to produce $\alpha_i$ (noting that $p_{\boldsymbol{\alpha},i} = 1$ if the $i^{\text{th}}$ tile attached without contention). Then $\Pr[\alpha] = \sum_{\boldsymbol{\alpha} \in A(\alpha)} \prod_{i=2}^{k} \frac{1}{|\partial\alpha_i|} p_{\boldsymbol{\alpha},i}$. Write $\mathcal{T}(\rho)$ to denote the random variable representing the producible, terminal assembly produced by $\mathcal{T}$ when using tile concentration assignment $\rho$.

## Problems

The general problem is this: given a shape $X \subset \mathbb{Z}^2$ (a connected, finite set), set the concentrations of tile types in some tile system $\mathcal{T}$ so that $\mathcal{T}$ is likely to create a terminal assembly with shape $X$, or "close to it." We now state formal problems that are variations on this theme. The first four problems use "concentration programming": varying the concentrations of tile types in a single tile system $\mathcal{T}$ to get it to assemble different shapes. The last two problems concern a tile system that only does one thing — assemble a line of a desired expected length — because in this setting we will require all concentrations to be equal. However, the tile system uses randomized self-assembly to do this with far fewer tile types than are needed to accomplish the same task in a deterministic tile system.

The first three problems concern the self-assembly of squares, and the problems are listed in order of increasing difficulty. The first asks for a square with a desired expected width, the second for a guarantee that the actual width is likely to be *close* to the expected width, and finally, for a guarantee that the actual width is likely to be *exactly* the expected width.

Formally, design a tile system $\mathcal{T} = (T, \sigma, \tau)$ such that, for any $n \in \mathbb{Z}^+$, there exists a tile concentration assignment $\rho : T \to [0, \infty)$ such that...

**Problem 1.** ... dom $\mathcal{T}(\rho)$ is a square with expected width $n$.

**Problem 2.** ... with probability at least $1 - \delta$, dom $\mathcal{T}(\rho)$ is a square whose width is between $(1 - \epsilon)n$ and $(1 + \epsilon)n$.

**Problem 3.** ... with probability at least $1 - \delta$, dom $\mathcal{T}(\rho)$ is a square of width $n$.

The next problem generalizes the previous problems to arbitrary shapes, while making one relaxation: allowing a scaled-up version of a shape to be assembled instead of the exact shape. Formally, for $c \in \mathbb{Z}^+$ and shape $S \subset \mathbb{Z}^2$ (finite and connected), define $S^c = \{ \ (x, y) \in \mathbb{Z}^2 \mid (\lfloor x/c \rfloor, \lfloor y/c \rfloor) \in S \ \}$ to be $S$ *scaled by factor* $c$.

**Problem 4.** Let $\delta > 0$. Design a tile system $\mathcal{T} = (T, \sigma, \tau)$ such that, for any shape $S \subset \mathbb{Z}^2$, there exists a tile concentration assignment $\rho : T \to [0, \infty)$ and $c \in \mathbb{Z}^+$ so that, with probability at least $1 - \delta$, dom $\mathcal{T}(\rho)$ is $S^c$.

It is easy to see that for a deterministic tile system to assemble a length $n$, height 1 line requires $n$ tile types. The next problem concerns using randomization to reduce the number of tile types required, subject to the constraint that all tile type concentrations are equal. (Without this constraint, a solution to Problem 1 would trivially be a solution to the next problem, with an optimal $O(1)$ tile types, but since the solution to Problem 1 uses different tile type concentrations to achieve its goal, it cannot be used directly for this purpose).

**Problem 5.** Let $n \in \mathbb{Z}^+$. Design a tile system $\mathcal{T} = (T, \sigma, \tau)$ such that, with tile concentration assignment $\rho : T \to [0, \infty)$ defined by $\rho(t) = 1$ for all $t \in T$, dom $\mathcal{T}(\rho)$ is a height 1 line of expected length $n$.

As with the case of concentration programming, it is desirable for the line to have length likely to be close to its expected length.

**Problem 6.** Let $n \in \mathbb{Z}^+$ and $\delta, \epsilon > 0$. Design a tile system $\mathcal{T} = (T, \sigma, \tau)$ such that, with tile concentration assignment $\rho : T \to [0, \infty)$ defined by $\rho(t) = 1$ for all $t \in T$, dom $\mathcal{T}(\rho)$ is a height 1 line whose length is between $(1-\epsilon)n$ and $(1+\epsilon)n$ with probability at least $1 - \delta$.

# Key Results

The solutions to Problems 1–4 use temperature 2 tile systems. The solutions to Problems 5 and 6 use a temperature 1 tile system (there is no need for cooperative binding in one dimension).

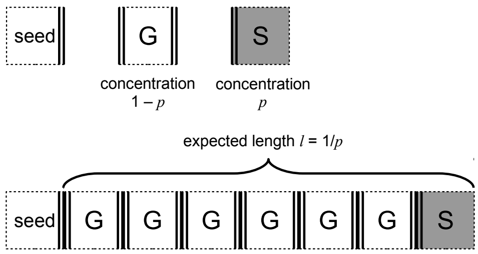


**Fig. 1.** A randomized temperature $\tau = 2$ tile system that can grow a line of any desired expected length $l$ by setting $p = \frac{1}{l}$. Two tiles compete nondeterministically to bind to the right of the line (using strength 2 glues, indicated by double black lines), one of which stops the growth, while the other continues, giving the length of the line (not counting the seed) a geometric distribution with expected value $l$.

Figure 1 shows a simple tile system with three tile types that can grow a line of any desired expected length to the right of the seed tile; this is the basis for the solutions to Problems 1, 2, 3, and 4. The length of the line has a geometic distribution, with expected value controlled by the ratio of the concentrations of $G$ and $S$. Figure 2 shows the solution to Problem 1, due to Becker, Remilá, and Rapaport [1]. It is essentially the tile system from Figure 1 (tile types $A$ and $B$ are analogous to $G$ and $S$ in Figure 1) augmented with a constant number of extra tiles that can assemble the square to be as high as the line is long.

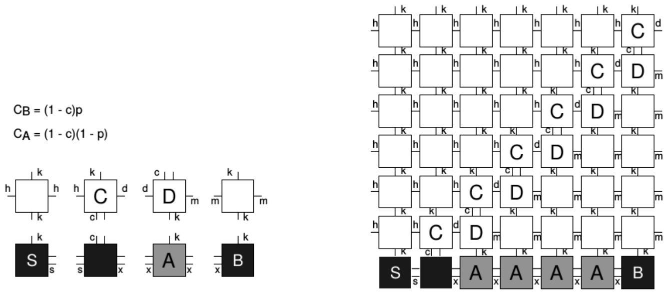


**Fig. 2.** A tile system that grows a square of any desired expected width. Figure taken from [4]; strength 2 glues are indicated by two lines between the tiles. The seed is labeled $S$, and $C_A$ and $C_B$ respectively represent the concentrations of $A$ and $B$. $p$ is used the same way as in Figure 1, and $c$ represents total concentation of all other tile types, since [4] assumed that concentrations of all tile types must sum to 1.

Kao and Schweller [4] showed a solution to Problem 2, and Doty [3] improved their construction to show a solution to Problem 3. Here, we describe only the latter construction, since the two share similar ideas, and the latter construction solves both problems.

Figure 3 shows an improvement to the tile system of Figure 1, which will be the starting point for the solution. It also can grow a line of any desired expected length. However, by using multiple independent "stages" of growth, each stage having a geometric distribution, the resulting assembly is more likely to have a length that is close to its expected length. More tile types are needed for more stages, but only a constant number of stages are required.

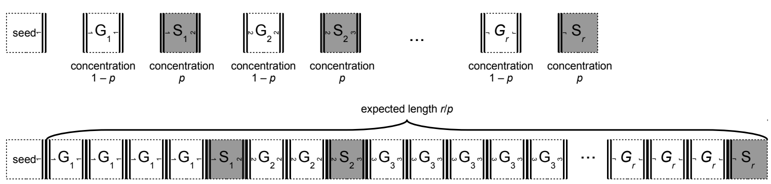


**Fig. 3.** A tile system that grows a line of a given length with greater precision than in Figure 1. $r$ stages each have expected length $1/p$, making the expected total length $r/p$, but more tightly concentrated about that expected length than in the case of one stage.

In particular, if the expected length is chosen to be midway between any two consecutive powers of two, i.e., midway in the interval $[2^{a-1}, 2^a)$ for arbitrary $a \in \mathbb{N}$, with $r = 113$ stages, the probability is at most 0.0025 that the actual length is outside the interval $[2^{a-1}, 2^a)$. So although the length is not controlled with exact precision, the number of bits needed to represent the length is controlled with exact precision (with high probability), using a constant number of tile types.

Figure 4 shows a tile system $\mathcal{T}$ with the following property: for any bit string $s$ (equivalently, any natural number $m$ if we assume the most significant bit of $s$ is 1), there is a tile concentration assignment that causes $\mathcal{T}$ to grow an assembly of height $O(\log m)$, width $O(m^2)$, such that the tile types in the upper-right corner of the assembly encode $s$. The bottom row is the tile system from Figure 3, with identical strength 2 glues on the north of the tiles (other than the final stop tile on the right).

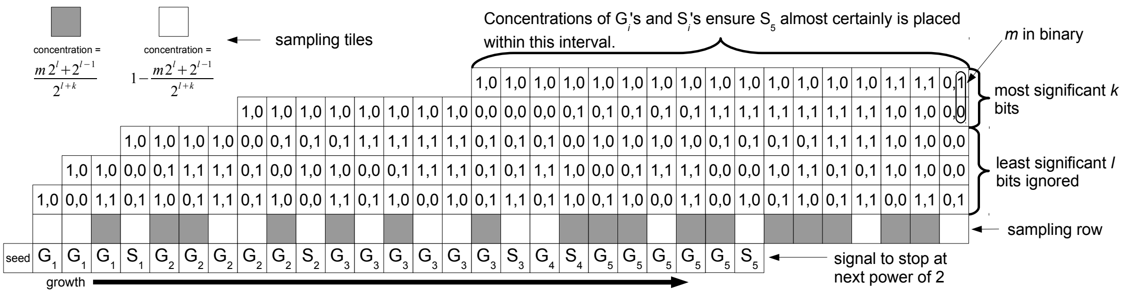


**Fig. 4.** Computing the binary string 10 (equivalently, the natural number $m = 2$) from tile concentrations. For brevity, glue strengths and labels are not shown. Each column increments the primary counter, represented by the bits on the left of each tile, and each gray tile increments the sampling 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 $c = 1$.

Figure 5 shows a high-level overview of the entire tile system that assembles an $n \times n$ square, solving Problem 3. Using similar ideas to Figure 4, one can encode three different numbers $m_1, m_2, m_3 \in \mathbb{N}$ into the tile concentrations. We choose these numbers to be such that each $m_i = O(n^{1/3})$, and each of their binary expansions, interwoven into a single bit string, is the binary expansion of $n$. Then each tile at the upper right of Figure 4 encodes not one but three bits of $n$, or equivalently each encodes an octal digit of $n$. These bits are then used to assemble a counter that counts from $n$ down to 0 as it grows north, and a constant set of tiles (similar to Figure 2 expand this counter to grow about as far east as the counter grows north, creating an $n \times n$ square that surrounds the assembly of Figure 4. Since $m_i = O(n^{1/3})$, and the tiles of Figure 4 create a structure of height $O(\log m_i)$ and width $O(m_i^2) = O(n^{2/3})$, the square is sufficiently large to contain the tiles of Figure 4.

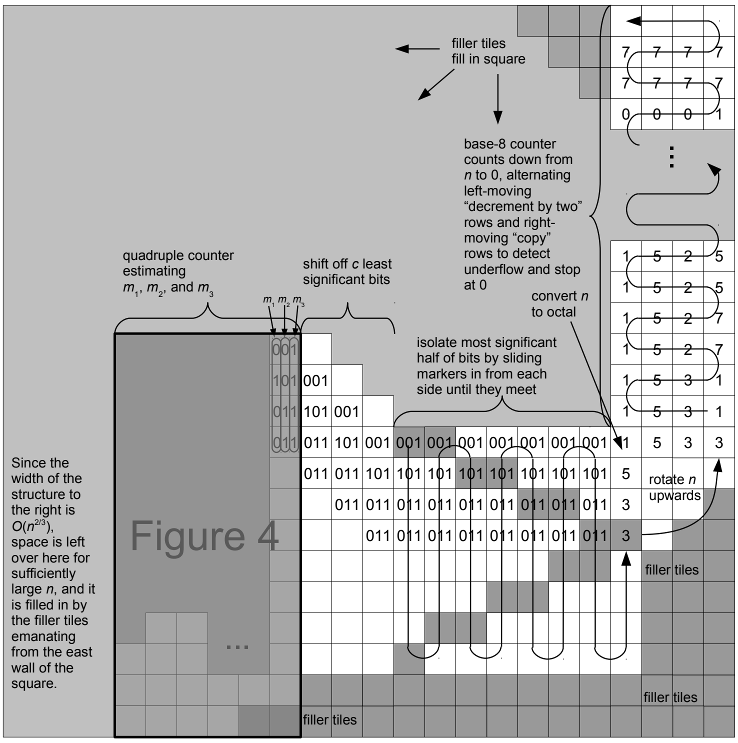


**Fig. 5.** High-level overview of the entire construction solving Problem 3, not at all to scale. For brevity, glue strengths and labels are not shown. The double counter number estimator of Figure 4 is embedded with two additional counters to create a quadruple counter estimating $m_1$, $m_2$, and $m_3$, shown as a box labeled as "Figure 4" 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. 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.

Finally, the tiles of Figure 4 are used in a different way to solve Problem 4, shown in Figure 6. Given a finite shape $S$, Soloveichik and Winfree [5] use an intricate construction of a "seed block" that "unpacks," from a set of tile types that depend on $S$, a single-tape Turing machine program $\pi \in \{0,1\}^*$ that outputs a binary string $\mathrm{bin}(S)$ representing a list of the coordinates of $S$.

The width of the seed block is then $c$, chosen to be large enough to do the unpacking, and also large enough to accommodate the simulation of $\pi$ by a tile set that simulates single-tape Turing machines. Once this seed block is in place, a tile set then assembles the scaled shape by carrying $\mathrm{bin}(S)$ through each block. The order in which blocks are assembled is determined by a spanning tree of $S$, so that any blocks with an ancestor relationship have a dependency, in that the ancestor must be (mostly) assembled before the descendant, whereas blocks without an ancestor relationship can potentially assemble in parallel.

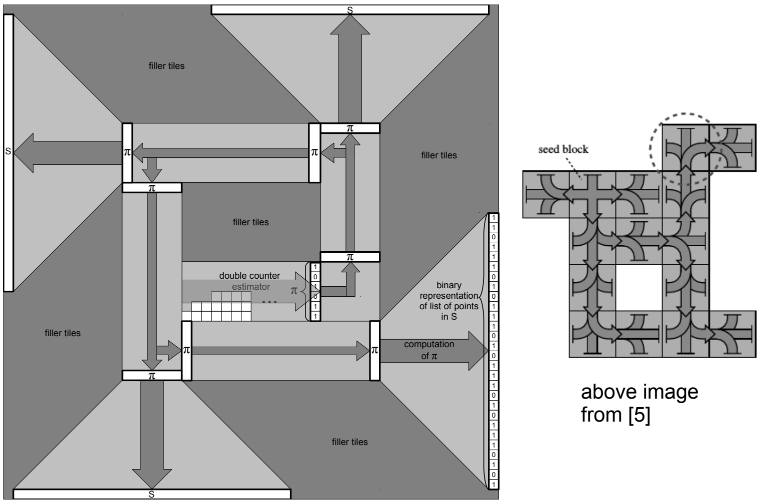


**Fig. 6.** On the left is the seed block used to replace the seed block of [5], from which the construction of [5] can assemble a scaled version of the shape $S$ (encoded by a binary string representing the list of coordinates, also labeled "$S$" in the figure). $S$ is output by the single-tape Turing machine program $\pi$. $\pi$ is estimated from tile concentrations as in Figure 4, then four copies of it are propagated to each side of the block, where it is executed in four rotated, but otherwise identical, computation regions. When completed, four copies of the binary representation of $S$ border the seed block, which is sufficient for the construction of [5] to assemble a scaled version of $S$ using a spanning tree of $S$ as shown on the right.

We replace the seed block tiles of [5], which depend on $S$, with a single tile system that produces the program $\pi$ from tile concentrations, and use the remainder of the tile set of [5] unchanged. This is illustrated in Figure 6. Choose $c$ to be sufficiently large that $\pi$ can be simulated within the trapezoidal region of the $c \times c$ block of Figure 6, and also sufficiently large that the construction of Figure 4 has sufficient room to estimate the binary string $\pi$ from tile concentrations in the center region (the "double counter estimator") of Figure 6. Once this is done, the construction of [5] can take over and assemble the entire scaled shape $S^c$. The portion of the construction of [5] that achieves this is a constant-size tile set, so combined with the presented construction remains constant. This solves Problem 4.

Finally, Problems 5 and 6 have solutions due to Chandran, Gopalkrishnan, and Reif [2], which we now explain intuitively (the actual analysis is a bit trickier but is close to the following intuitive argument). Figure 7 shows an example of a solution to Problem 5 for the case of expected length $n = 92$.

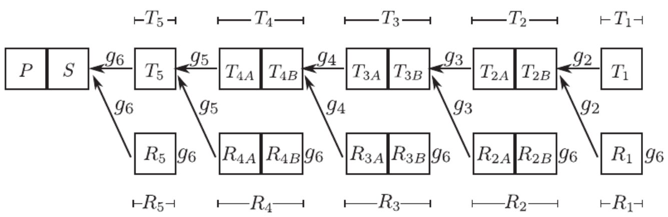


**Fig. 7.** Example of solution to Problem 5 for the case of expected length 92.

Each $T_{iB}$ tile type has an east glue, $g_i$, that matches two tile types $T_{(i-1)A}$ and $R_{(i-1)A}$. There are $O(\log n)$ "stages" (5 stages in this case). Each stage has probability $\frac{1}{2}$ to either decrement the stage or reset back to the highest stage. The number $n$ is programmed into the system by choosing each stage to have either 1 or 2 tiles. Given that we are in stage $i$, to make it from stage $i$ to stage 1 without resetting means that $i$ consecutive unbiased coin flips must come up "heads", which we expect to take $2^i$ flips before happening. Thus we expect stage $i$ to appear $2^i$ times; this means that stage $i$'s expected contribution to the total length is either $2^i$ or $2 \cdot 2^i$, depending on whether it has 1 or 2 tiles. The reason this works to encode arbitrary natural numbers $n$ is that every natural number can be expressed as $n = \sum_{i=0}^{\approx \log n} b_i 2^i$, where $b_i \in \{1, 2\}$. Since there are a constant number of tile types per stage, this implies that the number of tile types required is $O(\log n)$.

This solves Problem 5. To solve Problem 6, it suffices to concatenate $k$ independent assemblies of the kind shown in Figure 7, where $k$ is a constant that, if chosen sufficiently large based on $\delta$ (the desired error probability), solves Problem 6 since it increases the number of tile types required. In addition to proving that this works, Chandran, Gopalkrishnan, and Reif [2] also show a more complex construction with even sharper bounds on the probability that the length differs very much from its expected value.

## Open problems

The construction resolving Problem 3 shows that for every $\delta, \epsilon > 0$, a tile set exists such that, for every $n \in \mathbb{N}$, appropriately programming the tile concentrations results in the self-assembly of a structure of size $O(n^\epsilon) \times O(\log n)$ whose rightmost tiles represent the value $n$ with probability at least $1 - \delta$. (In the tile system described, $\epsilon = 2/3$, and it could be made arbitrarily close to 0 by estimating more than 3 numbers at once.) Is this optimal?

Formally, say that a tile assembly system $\mathcal{T} = (T, \sigma, 2)$ is *$\delta$-concentration programmable* (for $\delta > 0$) if there is a (total) computable function $r : \mathcal{A}_\Box[\mathcal{T}] \to \mathbb{N}$ (the *representation function*) such that, for each $n \in \mathbb{N}$, there is a tile concentration assignment $\rho : T \to [0, \infty)$ such that $\Pr[r(\mathcal{T}(\rho)) = n] \geq 1 - \delta$. In other words, $\mathcal{T}$, programmed with concentrations $\rho$, almost certainly self-assembles a structure that "represents" $n$, according to the representation function $r$, and such a $\rho$ can be found to create a high-probability representation of *any* natural number.

*Question 1.* Is the following statement true? For each $\delta > 0$, there is a tile assembly system $\mathcal{T}$ and a representation function $r : \mathcal{A}_\Box[\mathcal{T}] \to \mathbb{N}$ such that $\mathcal{T}$ is $\delta$-concentration programmable and, for each $\epsilon > 0$ and all but finitely many $n \in \mathbb{N}$, $\Pr[|\mathrm{dom}\ \mathcal{T}(\rho)| < n^\epsilon] \geq 1 - \delta$. If so, what is the smallest bound that can be written in place of $n^\epsilon$?

## Recommended Reading


1. Florent Becker, Ivan Rapaport, and Eric Rémila. Self-assembling classes of shapes with a minimum number of tiles, and in optimal time. In *FSTTCS 2006: Foundations of Software Technology and Theoretical Computer Science*, pages 45–56, 2006.
2. Harish Chandran, Nikhil Gopalkrishnan, and John H. Reif. Tile complexity of linear assemblies. *SIAM Journal on Computing*, 41(4):1051–1073, 2012. Preliminary version appeared in ICALP 2009.
3. David Doty. Randomized self-assembly for exact shapes. *SIAM Journal on Computing*, 39(8):3521–3552, 2010. Preliminary version appeared in FOCS 2009.
4. Ming-Yang Kao and Robert T. Schweller. Randomized self-assembly for approximate shapes. In *ICALP 2008: International Colloqium on Automata, Languages, and Programming*, volume 5125 of *Lecture Notes in Computer Science*, pages 370–384. Springer, 2008.
5. David Soloveichik and Erik Winfree. Complexity of self-assembled shapes. *SIAM Journal on Computing*, 36(6):1544–1569, 2007. Preliminary version appeared in DNA 2004.
6. Erik Winfree. *Algorithmic Self-Assembly of DNA*. PhD thesis, California Institute of Technology, June 1998.