Fast Algorithmic Self-assembly of Simple Shapes Using Random Agitation

Ho-Lin Chen[1,⋆], David Doty[2,⋆⋆], Dhiraj Holden[2,⋆⋆⋆]
Chris Thachuk[2,†], Damien Woods[2,‡], and Chun-Tao Yang[1,§]

[1] Dept. of Electrical Engineering, National Taiwan University, Taipei 106, Taiwan
[2] California Institute of Technology, Pasadena, CA 91125, USA
holinchen@ntu.edu.tw, {ddoty,dholden,thachuk,woods}@caltech.edu,
havachoice@gmail.com

Abstract. We study the power of uncontrolled random molecular movement in a model of self-assembly called the nubots model. The nubots model is an asynchronous nondeterministic cellular automaton augmented with rigid-body movement rules (push/pull, deterministically and programmatically applied to specific monomers) and random agitations (nondeterministically applied to every monomer and direction with equal probability all of the time). Previous work on nubots showed how to build simple shapes such as lines and squares quickly—in expected time that is merely logarithmic of their size. These results crucially make use of the programmable rigid-body movement rule: the ability for a single monomer to push or pull large objects quickly, and only at a time and place of the programmers' choosing. However, in engineered molecular systems, molecular motion is largely uncontrolled and fundamentally random. This raises the question of whether similar results can be achieved in a more restrictive, and perhaps easier to justify, model where uncontrolled random movements, or agitations, are happening throughout the self-assembly process and are the only form of rigid-body movement. We show that this is indeed the case: we give a polylogarithmic expected time construction for squares using agitation, and a sublinear expected time construction to build a line. Such results are impossible in an agitation-free (and movement-free) setting and thus show the benefits of exploiting uncontrolled random movement.

1 Introduction

Every molecular structure that has been self-assembled in nature or in the lab was assembled in conditions (above absolute zero) where molecules are vibrating

⋆ Supported by NSC grant number 101-2221-E-002-122-MY3.
⋆⋆ Supported by National Science Foundation grants 0832824 & 1317694 (The Molecular Programming Project), CCF-1219274, CCF-1162589.
⋆⋆⋆ Supported by NSF grant CCF-1219274.
† Supported by NSF grant CCF/HCC-1213127.
‡ Supported by National Science Foundation grants 0832824 & 1317694 (The Molecular Programming Project), CCF-1219274, CCF-1162589.
§ Supported by NSC grant number 101-2221-E-002-122-MY3.

S. Murata and S. Kobayashi (Eds.): DNA 2014, LNCS 8727, pp. 20–36, 2014.

relative to each other, randomly bumping into each other via Brownian motion, and often experiencing rapid uncontrolled fluid flows. It makes sense then to study a model of self-assembly that includes, and indeed allows us to exploit and program, such phenomena. It is a primary goal of this paper to show the power of self-assembly under such conditions.

In the theory of molecular-scale self-assembly, millions of simple interacting components are designed to autonomously stick together to build complicated shapes and patterns. Many models of self-assembly are cellular automata-like crystal growth models, such as the abstract tile assembly model [9]. Indeed this and other such models have given rise to a rich theory of self-assembly [5,8,10]. In biological systems we frequently see much more sophisticated growth processes, where self-assembly is combined with active molecular motors that have the ability to push and pull large structures around. For example, during the gastrulation phase of the embryonic development of the model organism *Drosophila melanogaster* (a fly) large-scale (100s of micrometers) rearrangements of the embryo are effected by thousands of (nanoscale) molecular motors working together to rapidly push and pull the embryo into a final desired shape [4,7]. We wish to understand, at a high level of abstraction, the ultimate computational capabilities and limitations of such molecular scale rearrangement and growth.

The nubots model of self-assembly, put forward in [11], is an asynchronous nondeterministic cellular automaton augmented with non-local rigid-body movement. Unit-sized monomers are placed on a 2D hexagonal grid. Monomers can undergo state changes, appear, and disappear, using local cellular-automata style rules. However, there is also a non-local aspect to the model, a kind of rigid body movement that comes in two forms: movement rules and random agitations. A movement rule r, consisting of a pair of monomer states and two unit vectors, is a programatic way to specific unit-distance translation of a set of monomers in one step: if two adjacent monomers on the grid have states A and B and are in a prescribed orientation, then we may try to apply r so that one of A or B *moves* unit distance in a prescribed direction relative to the other. The rule r is applied in a rigid-body fashion: if A is to move right, it pushes anything immediately to its right and pulls any monomers that are bound to its left (roughly speaking), which in turn push and pull other monomers. The rule may not be applicable if it is blocked (i.e. if movement of A would force B to also move), which is analogous to the fact that an arm can not push its own shoulder. The other form of movement in the model is called *agitation*: at every point in time, every monomer on the grid may move unit distance in any of the six directions, at unit rate for each (monomer, direction) pair. An agitating monomer will push or pull any monomers that it is adjacent to, in a way that preserves rigid-body structure. Unlike movement, agitations are never blocked. Rules are applied asynchronously and in parallel in the model. Taking its time model from stochastic chemical kinetics, a nubots system evolves as a continuous time Markov process.

In summary, there are two kinds of movement in the model: (a) a *movement rule* is applied only to a pair of monomers with the prescribed states and orientation, and then causes the movement of one of these monomers (and other pushed/pulled monomers) but not the other, whereas (b) *agitations* are always applicable at every time instant, in every direction and to every monomer throughout the grid.

In previous work, the movement rule was exploited to show that nubots is very efficient in terms of its computational ability to quickly build complicated shapes and patterns. Agitation was treated as something to be robust against (i.e. the constructions in [11,2] work both with and without agitation), which seems like a natural requirement when building structures in a molecular-scale environment. However, it was left open as to whether the kind of results achieved with movement could be achieved without movement, but by exploiting agitation [2]. In other words, it was left open as to whether augmenting a cellular automaton with an uncontrolled form of random rigid-body movement would facilitate functionality that is impossible without it. Here we show this is the case.

Agitation, and the movement rule, are defined in such a way that larger objects move faster, and this is justified by imagining that we are self-assembling rigid-body objects in a nanoscale environment where there are uncontrolled and turbulent fluid flows in all directions interacting with each monomer at unit rate per monomer. It remains as an interesting open research direction to look at the nubots model but with a slower rate model for agitation and movement, specifically where we hold on to the notion of rigid body movement and/or agitation but where bigger things move slower, as seen in Brownian motion for example. Independent of the choice of rate model, one of our main motivations here is to understand what can be done with *asynchronous, distributed and parallel* self-assembly with *rigid body motion*: the fact that our systems work in a parallel fashion is actually more important to us than the fact they are fast. It is precisely this engineering of distributed asynchronous molecular systems that interests us.

The nubots model is related to, but distinct from, a number of other self-assembly and robotics models as described in [11]. Besides the fact that biological systems make extensive use of molecular-scale movements and rearrangements, in recent years we have seen the design and fabrication of a number of molecular-scale DNA motors [1] and active self-assembly systems which also serve to motivate our work, details of which can be found in previous papers on nubots [11,2].

1.1 Results and Future Work

Agitation nubots denotes the nubots model without the movement rule and with agitation (see Section 2 for formal definitions). The first of our two main results shows that agitation can be exploited to build a large object exponentially quickly:

Theorem 1. *There is a set of nubots rules* $\mathcal{N}_{\text{square}}$*, such that for all* $n \in \mathbb{N}$*, starting from a line of* $\lfloor \log_2 n \rfloor + 1$ *monomers, each in state* 0 *or* 1*,* $\mathcal{N}_{\text{square}}$ *in*

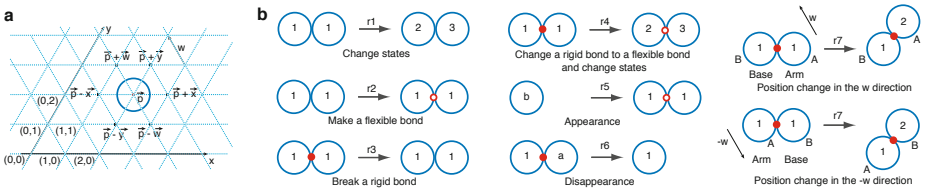

Fig. 1. Overview of nubots model. (a) A nubot configuration showing a single nubot monomer on the triangular grid. (b) Examples of nubot monomer rules. Rules r1-r6 are local cellular automaton-like rules, whereas r7 effects a non-local movement. A flexible bond is depicted as an empty red circle and a rigid bond is depicted as a solid red disk. Rules and bonds are described more formally in Section 2.

the agitation nubots model assembles an $n \times n$ square in $O(\log^2 n)$ expected time, $n \times n$ space and $O(1)$ monomer states.

The proof is in Section 4. Our second main result shows that we can achieve sublinear expected time growth of a length n line in only $O(n)$ space:

Theorem 2. *There is a set of nubots rules $\mathcal{N}_{\text{line}}$, such that for any $\epsilon > 0$, for sufficiently large $n \in \mathbb{N}$, starting from a line of $\lfloor \log_2 n \rfloor + 1$ monomers, each in state 0 or 1, $\mathcal{N}_{\text{line}}$ in the agitation nubots model assembles an $n \times 1$ line in $O(n^{1/3+\epsilon})$ expected time, $n \times 5$ space and $O(1)$ monomer states.*

The proof is in Section 5. Lines and squares are examples of fundamental components for the self-assembly of arbitrary computable shapes and patterns in nubots [11,2,3] and other self-assembly models [5,8].

Our work here suggests that random agitations applied in an uncontrolled fashion throughout the grid are a powerful resource. However, are random agitations as powerful as the programable and more deterministic movement rule used in previous work on nubots [11,2]? In other words can agitation *simulate* movement? More formally, is it the case that for each nubots program $\mathcal{N}$, there is an agitation nubots program $\mathcal{A}_{\mathcal{N}}$, that acts just like $\mathcal{N}$ but with some $m \times m$ scale-up in space, and a k factor slowdown in time, where m and k are (constants) independent of $\mathcal{N}$ and its input? This question is inspired by the use of simulations in tile assembly as a method to classify and separate the power of self-assembly systems, for more details see [6,10]. It would also be interesting to know whether the full nubots model, and indeed the agitation nubots model, are intrinsically universal [6,10]. That is, is there a single set of nubots rules that simulate any nubots system? Is there a single set of agitation nubots rules that simulate any agitation nubots system? Here the scale factor m would be a function of the simulated system $\mathcal{N}$. As noted in the introduction, it remains as an interesting open research direction to look at the nubots model but with a slower rate model for agitation and movement, as seen in Brownian motion, for example.

2 The Nubots Model

In this section we formally define the nubots model. Figure 1 gives an overview of the model and rules. Figure 3 shows a simple example construction using only local rules, while Figure 2 gives two examples of agitation.

The model uses a two-dimensional triangular grid with a coordinate system using axes x and y as shown in Figure 1(a). A third axis, w, is defined as running through the origin and $\overrightarrow{w} = -\overrightarrow{x} + \overrightarrow{y} = (-1, 1)$, but we use only the x and y coordinates to define position. The *axial directions* $\mathcal{D} = \{\pm\overrightarrow{x}, \pm\overrightarrow{y}, \pm\overrightarrow{w}\}$ are the unit vectors along axes x, y, w. A pair $\overrightarrow{p} \in \mathbb{Z}^2$ is called a *grid point* and has the set of six *neighbors* $\{\overrightarrow{p} + \overrightarrow{u} \mid \overrightarrow{u} \in \mathcal{D}\}$. Let S be a finite set of monomer states. A nubot *monomer* is a pair $X = (s_i, p(X))$ where $s_i \in S$ is a state and $p(X) \in \mathbb{Z}^2$ is a grid point. Two monomers on neighboring grid points are either connected by a *flexible* or *rigid* bond, or else have no bond (called a *null* bond). Bonds are described in more detail below. A *configuration* C is a finite set of monomers along with all of the bonds between them (unless otherwise stated a configuration consists of *all* of the monomers on the grid and their bonds).

One configuration *transitions* to another either via the application of a *rule* that acts on one or two monomers, or by an *agitation*. For a rule $r = (s1, s2, b, \overrightarrow{u}) \rightarrow (s1', s2', b', \overrightarrow{u}')$, the left and right sides of the arrow respectively represent the contents of the two monomer positions before and after the application of r. Specifically, $s1, s2, s1', s2' \in S \cup \{\mathsf{empty}\}$ are monomer states where empty denotes lack of a monomer, $b, b' \in \{\mathsf{flexible}, \mathsf{rigid}, \mathsf{null}\}$ are bond types, and $\overrightarrow{u}, \overrightarrow{u}' \in \mathcal{D}$ are unit vectors. b is a bond type between monomers with state $s1$ and $s2$, and $\overrightarrow{u} \in \mathcal{D}$ is the relative position of a monomer with state $s2$ to a monomer with state $s1$ (likewise for $b', s1', s2', \overrightarrow{u}'$). At most one of $s1, s2$ is empty (we disallow spontaneous generation of monomers from empty space). If $\mathsf{empty} \in \{s1, s2\}$ then $b = \mathsf{null}$, likewise if $\mathsf{empty} \in \{s1', s2'\}$ then $b' = \mathsf{null}$.

A rule either does not or does involve movement (translation). First, in the case of no movement we have $\overrightarrow{u} = \overrightarrow{u}'$. Thus we have a rule of the form $r = (s1, s2, b, \overrightarrow{u}) \rightarrow (s1', s2', b', \overrightarrow{u})$, where the monomer pair may *change state* ($s1 \neq s1'$ and/or $s2 \neq s2'$) and/or *change bond* ($b \neq b'$), examples are shown in Figure 1(b). If $s_i \in \{s1, s2\}$ is empty and s_i' is not, then the rule is said to induce the *appearance* of a new monomer at the empty location. If one or both monomer states go from non-empty to empty, the rule induces the *disappearance* of one or both monomers. Second, in the case of a movement rule, $\overrightarrow{u} \neq \overrightarrow{u}'$ and the rule has a specific form defined in [11,2]. Movement rules are not used in the *agitation nubots* model studied in this paper, and so their definition may be ignored by the reader. A rule is only applicable in the orientation specified by $\overrightarrow{u}$.

To define agitation we introduce some notions. Let $\overrightarrow{v} \in \mathcal{D}$ be a unit vector. The $\overrightarrow{v}$-boundary of a set of monomers S is defined to be the set of grid points outside of S that are unit distance in the $\overrightarrow{v}$ direction from monomers in S.

Definition 3 (Agitation set). *Let C be a configuration containing monomer A, and let $\overrightarrow{v} \in \mathcal{D}$ be a unit vector. The* agitation set $\mathcal{A}(C, A, \overrightarrow{v})$ *is defined to be the smallest monomer set in C containing A that can be translated by $\overrightarrow{v}$ such that:*

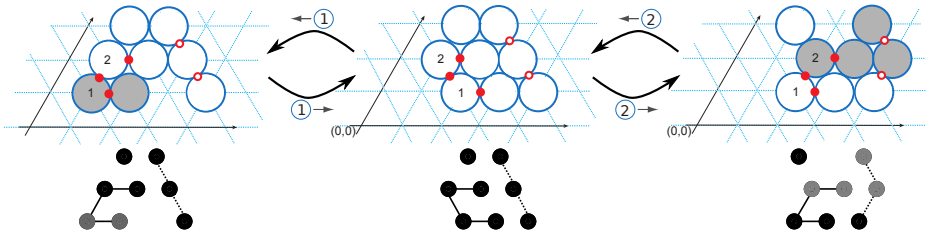

Fig. 2. Top: Example agitations. Starting from the centre configuration, there are 48 possible agitations (8 monomers, 6 directions each), any one of which is chosen with equal probability 1/48. The right configuration results from the agitation of the monomer at position $(1, 2)$ in the direction $\rightarrow$, starting from the centre configuration. The left configuration results from the agitation of the monomer at position $(2, 1)$ in the direction $\leftarrow$, starting from the centre configuration. The shaded monomers are the *agitation set*—the set of monomers that are moved by the agitation—when beginning from the centre configuration. Bottom: simplified ball-and-stick representation of the monomers and their bonds, which is used in a number of other figures.

(a) monomer pairs in C that are joined by rigid bonds do not change their relative position to each other, (b) monomer pairs in C that are joined by flexible bonds stay within each other's neighborhood, and (c) the $\overrightarrow{v}$-boundary of $\mathcal{A}(C, A, \overrightarrow{v})$ contains no monomers.

We now define agitation. An *agitation* step acts on an entire configuration C as follows. A monomer A and unit vector $\overrightarrow{v}$ are selected uniformly at random from the configuration of monomers C and the set of six unit vectors $\mathcal{D}$ respectively. Then, the agitation set $\mathcal{A}(C, A, \overrightarrow{v})$ of monomers (Definition 3) moves by vector $\overrightarrow{v}$.

Figure 2 gives two examples of agitation. Some remarks on agitation: It can be seen that for any non-empty configuration the agitation set is always non-empty. During agitation, the only change in the system configuration is in the positions of the constituent monomers in the agitation set, and all the monomer states and bond types remain unchanged. We let *agitation nubots* denote the nubots model without the movement rule. Agitation is intended to model movement that is not a direct consequence of a rule application, but rather results from diffusion, Brownian motion, turbulent flow or other uncontrolled inputs of energy.

An *assembly system* $T = (C_0, \mathcal{N})$ is a pair where C_0 is the initial configuration, and $\mathcal{N}$ is the set of rules. If configuration C_i transitions to C_j by some rule $r \in \mathcal{N}$, or by an agitation step, we write $C_i \vdash_{\mathcal{N}} C_j$. A *trajectory* is a finite sequence of configurations $C_1, C_2, \ldots, C_\ell$ where $C_i \vdash_{\mathcal{N}} C_{i+1}$ and $1 \leq i \leq \ell - 1$. An assembly system is said to *assemble* a shape or pattern if, starting from some initial configuration C_0, every trajectory evolves to the desired shape or pattern. An assembly system evolves as a continuous time Markov process. The rate for each rule application, and for each agitation step, is 1. If there are k applicable transitions for a configuration C_i (i.e. k is the sum of the number of rule and agitation steps that can be applied to all monomers), then the probability of any given transition being applied is $1/k$, and the time until the next transition is

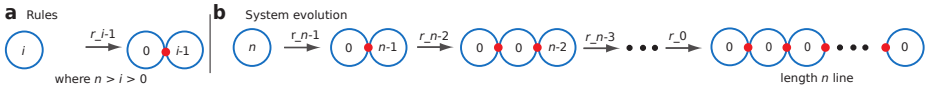

Fig. 3. A nubots system that slowly grows a length n line in $O(n)$ time, n monomer states, and using space $n \times 1$. (a) Rule set: $\mathcal{R}_n^{\text{slow line}} = \{r_i \mid r_i = (i, \mathsf{empty}, \mathsf{null}, \boldsymbol{x}) \rightarrow (0, i-1, \mathsf{rigid}, \boldsymbol{x}),$ where $n > i > 0\}$. (b) Starting from an initial configuration with a single monomer in state n, the system generates a length n line. Taken from [11].

applied is an exponential random variable with rate k (i.e. the expected time is $1/k$). The probability of a trajectory is then the product of the probabilities of each of the transitions along the trajectory, and the expected time of a trajectory is the sum of the expected times of each transition in the trajectory. Thus, $\sum_{t \in \mathcal{T}} \Pr[t]\text{time}(t)$ is the expected time for the system to evolve from configuration C_i to configuration C_j, where $\mathcal{T}$ is the set of all trajectories from C_i to any configuration isomorphic (up to translation) to C_j, that do not pass through any other configuration isomorphic to C_j, and time(t) is the expected time for trajectory t.

The complexity measure *number of monomers* is the maximum number of monomers that appears in any configuration. The *number of states* is the total number of distinct monomer states that appear in the rule set. *Space* is the maximum, over the set of all reachable configurations, of the minimum-sized $l \times w$ rectangle (on the hex grid) that, up to translation, contains all monomers in the configuration.

2.1 Example: A Simple, But Slow, Method to Build a Line

Figure 3, from [11], shows a simple method to build a length n line in expected time n, using $O(n)$ monomer states. Here, the program is acting as an asynchronous cellular automata and is *not* exploiting the ability of a large set of monomers to quickly move via agitation. Our results show that we can do much better than this very slow and expensive (many states) method to grow a line.

3 Synchronization via Agitation

In this section we describe a fast method to synchronize the states of a line of monomers using agitation. Specifically, the *synchronization* problem is: given a length-m line of monomers that are in a variety of states but that all eventually reach some target state s, then after all m monomers have reached state s, communicate this fact to all m monomers in $O(\log m)$ expected time.

Lemma 4 (Synchronization). *A line of monomers of length $m \in \mathbb{N}$ can be synchronized (all monomers put into the same state) in $O(\log m)$ expected time, with $O(1)$ states, and in $m \times O(1)$ space.*

The proof is described in Figure 4 and its caption. The figure gives a synchronization routine that is used throughout our constructions. This is a modification of

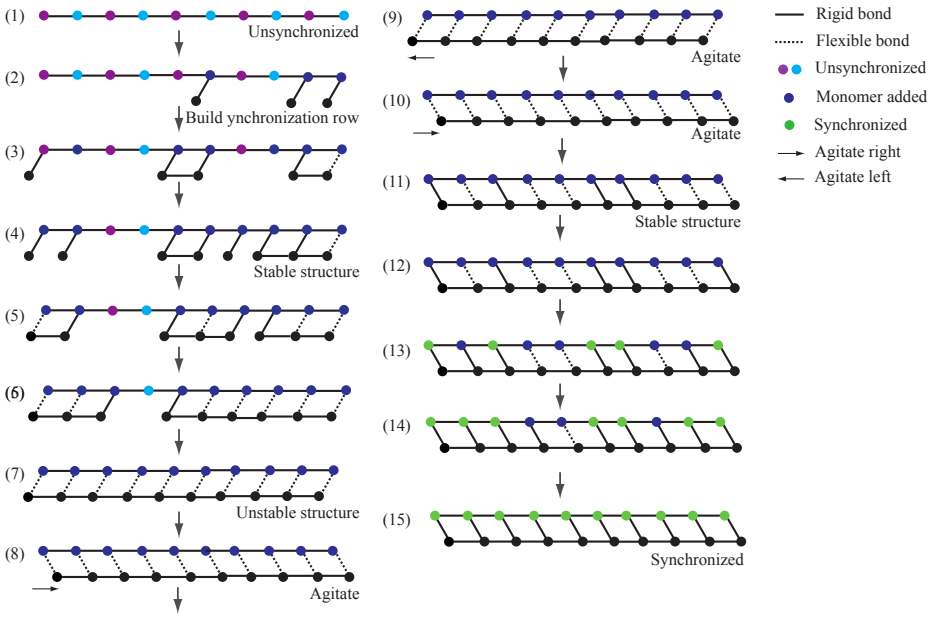

Fig. 4. Synchronization via agitation: a nubots construction to synchronize (or send a signal, or reach consensus) between n monomers in $O(\log n)$ expected time. Steps (1)–(6): build a row of monomers called the synchronization row. Rigid bonds are converted to flexible bonds in such a way that agitations do not change the relative position of monomers. A structure with this property is said to be *stable*. Specifically, monomers are added using rigid vertical bonds; new monomers join to left-right neighbours using rigid horizontal bonds; when a monomer is bound horizontally to both neighbours it makes its *vertical* bond flexible; monomers on the extreme left and right of the synchronization row are treated differently—their vertical bonds become flexible after joining any horizontal neighbour. This enforces that the entire structure is stable up until the final horizontal bond is added, and then the structure becomes unstable in such a way that the synchronization row can agitate left-right relative to the backbone row. Steps (7)–(10), the structure is not stable, and the synchronization row is free to agitate left and right relative to the backbone row. While agitating, the synchronization row spends half the time to the left, and half to the right, of the backbone row. However, whenever the synchronization row is to the right a rigid bond may form between any synchronization row monomer and the backbone monomer directly above, hence the first such bond forms in expected time $1/m$, where m is the length of the backbone. Then all bonds become rigid in $O(\log m)$ expected time, during which time (12)–(15) the backbone monomers change their state to the final *synchronized* state.

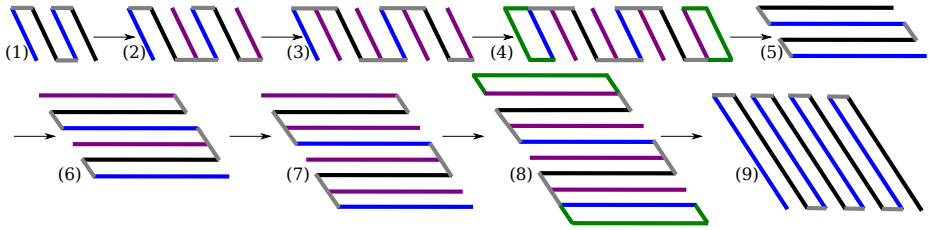

Fig. 5. An overview of the square doubling algorithm that grows an $m \times m$ zig-zag "comb" to a $2m \times 2m$ comb. (1) An initial $m \times m$ comb with vertical teeth, is (2) "half-doubled" to give a $\lfloor 1.5m \rfloor \times m$ comb, which is (3) again half-doubled to give a $2m \times m$ comb. (4)–(5) The internal bond structure is reconfigured to give a comb with horizontal teeth. (6)–(7) this comb is vertically doubled in size and then (8)–(9) reconfigured to give a $2m \times 2m$ comb with vertical teeth. The green lines indicate temporary synchronization rows that are used when reorientating the teeth of the comb.

the synchronization routine in [11], made to work with agitation instead of the movement rule.

4 Building Squares via Agitation

This section contains the proof of our first main result, Theorem 1.

Proof (Theorem 1). **Overview of Construction.** Figure 5 gives an overview of our construction. A binary string that represents $n \in \mathbb{N}$ in the standard way is encoded as a string x, of length $\ell = \lfloor \log_2 n \rfloor + 1$, of adjacent rigidly bound binary nubot monomers (each in state 0 or 1) placed somewhere on the hexagonal grid.

The leftmost of these monomers begins an iterated square-doubling process, that happens exactly ℓ times. Each iteration of this square-doubling process: reads the current most significant bit x_i of x, where $0 \leq i \leq \ell$, stores it in the state of a monomer in the top-left of the square and then deletes x_i. Then, if $x_i = 0$ it takes an $m \times m$ comb structure and doubles its size to give a $2m \times 2m$ comb structure, or if $x_i = 1$ it gives a $(2m+1) \times (2m+1)$ structure. We will prove that each square-doubling step takes $O(\log m)$ time. There are ℓ rounds of square-doubling, i.e. the number of input monomers ℓ act as a counter to control the number of iterations, and since $m \leq n$ throughout, the process completes in the claimed expected time of $O(\log^2 n)$. The main part of the construction, detailed below, lies in the details of how each doubling step works and an expected time analysis, and constitutes the remainder of the proof.

Doubling Construction. A single square-doubling consists of four phases: two horizontal "half-doublings" and two vertical half-doublings. Figure 5 gives an overview. Figure 6 gives the details of how we do the first of two horizontal half-doublings; more precisely, the figure shows how to go from an $m \times m$ structure to a structure of size $\lfloor 1.5m \rfloor \times m$. Assume we are at a configuration with m vertical comb teeth (Figure 6(1)) each of height m (plus some additional monomers).

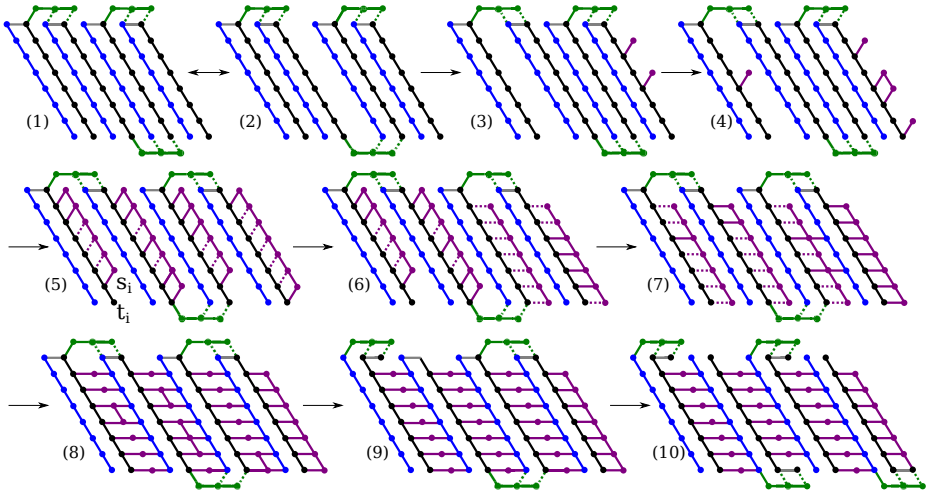

Fig. 6. The $m \times m$ to $\lfloor 1.5m \rfloor \times m$ horizontal half-doubling algorithm, for $m = 8$. This shows the details for step (1) to (2) of Figure 5. Monomer states are denoted using colours (bonds are also coloured for readability). Rigid bonds are solid, flexile bonds are dotted. See main text for details.

Teeth are numbered from the left $t_1, t_2, \ldots, t_m$. Each tooth monomer undergoes agitation. It can be seen in Figure 6(1)–(4), from the bond structure, that the only agitations that change the relative position of monomers are left or right agitations which move the green flexible bonds (depicted as dashed lines)—all other agitations move the entire structure without changing the relative positions of any monomers. Furthermore, left-right monomer agitations can create gaps between teeth t_i and t_{1+1} for even i only—for odd i, teeth t_i and t_{1+1} are rigidly bound. An example of a gap opening between tooth t_4 and tooth t_5 is shown in Figure 6(2). If a gap appears between teeth t_i and t_{1+1} then each of the m monomers in tooth t_i tries to attach a new purple monomer to its right (with a rigid bond, and each at rate 1), so attachment for any monomer to tooth i happens at rate m. (Note that the gap is closing and opening at some rate also—details in the time analysis.) After the first such purple monomer appears, the gap g_i, to the right of tooth t_i, is said to be "initially filled". For example, in Figure 6(4), gap g_2 is initially filled.

When gaps appear between teeth monomers, and then become initially filled, additional monomers are attached, asynchronously and in parallel. Monomers attaching to tooth t_i initially attach by rigid bonds as shown in Figure 6(4). As new monomers attach to t_i, they then attempt to bind to each other vertically, and after such a binding event they undergo a sequence of bond changes—see Figure 6(4)-(9). Specifically, let $s_{i,j}$ be the j^{th} monomer on the newly-forming "synchronization row" s_i adjacent to t_i. When the neighbors $s_{i,j-1}, s_{i,j+1}$ of monomer $s_{i,j}$ appear, then $s_{i,j}$ forms rigid bonds with them (at rate 1). After this, $s_{i,j}$ changes its rigid bonds to $t_{i,j}$ to flexible. The top and bottom monomers $s_{i,1}$, $s_{i,m}$ are special cases: their bonds to $t_{i,1}$, $t_{i,m}$ become flexible

after they have joined to their (single) neighbors $s_{i,2}$, $s_{i,m-1}$. Changing bonds in this order guarantees that only *after all monomers* of s_i have attached, and not before, the synchronization row s_i is free to agitate up and down relative to the tooth t_i (this is the same technique for building a synchronization row as described in Section 3). The new vertical synchronization row s_i is then free to agitate up and down relative to its left-adjacent tooth t_i. When $s_{i,j}$ is "down" relative to $t_{i,j}$ the horizontal bonds between $s_{i,j}$ and $t_{i,j}$ become rigid, at rate 1 per bond (Figure 6(6)–(7)). When the vertical synchronization of s_i is done, a message is sent from the top monomer $t_{i,m}$ of t_i (after its bond to $s_{i,m}$ becomes rigid) to the adjacent monomer at the top of the comb. This results in the formation of a horizontal synchronization row at the top of the structure. Using a similar technique, a horizontal synchronization row grows at the bottom of the structure. After all $2\lfloor 0.5m \rfloor$ such messages have arrived, and not before, the horizontal synchronization rows at the top and bottom of the (now) $\lfloor 1.5m \rfloor \times m$ comb change the last of their rigid (vertical) bonds to flexible and those synchronization rows are free to agitate left/right and then lock into position, signaling to all monomers along their backbone that the first of the four half-doublings of the comb has finished.

The system prepares for the next horizontal half-doubling which will grow the $\lfloor 1.5m \rfloor \times m$ comb to be an $2m \times m$ comb. The bonds at the top and bottom horizontal synchronization rows reconfigure themselves (preserving connectivity of the overall structure—see the description of reconfiguration below) in such a way as to build the gadgets needed for the next half-doubling. (Specifically, we want to now double teeth t_i for odd $i \leq m$.) The construction proceeds similarly to the first half-doubling, except for the following change. After tooth synchronization row s_1 has synchronized, tooth t_1 grows a vertical synchronization row to its left, and after s_m has synchronized, tooth t_m grows a vertical synchronization row to its right (Figure 5(4)). These two synchronization rows are used to set-up the bond structure for the next stage of the construction (where we will reconfigure the entire comb so that the teeth are horizontal).

This covers the case of the input bit being 0. Otherwise, if the input bit is 1, adding an extra tooth can be done using the single vertical synchronization row on the right—it reconfigures itself to have the bond structure of a tooth and then grows a new vertical synchronization row.

Reconfiguration Construction. Next we describe how the comb with vertical teeth is reconfigured to have horizontal teeth, as in Figure 5(4)–(5). After synchronization row s_i has synchronized, each monomer $s_{i,j}$ in s_i already has a rigid horizontal bond to monomer $t_{i,j}$. After both s_i and s_{i+1} have synchronized, for all j, monomers $s_{i,j}$ and $t_{i+1,j}$ bond using a horizontal rigid bond (at rate 1) for each pair $(s_{i,j}, t_{i+1,j})$. Monomers t_i and s_i then delete their vertical rigid bonds in such a way that preserves the overall connectivity of the structure. (For these bond reconfigurations we are simply using local—asynchronous cellular automaton style—rules that preserves connectivity. This trick has been used in previous nubots constructions in [11,2].) This leads to a bond structure similar to that in Figure 6(10) both with roughly twice the number of horizon-

tal purple bonds: i.e. for each j, $1 \leq j \leq m$, there is now a horizontal straight line of purple bonds from the jth monomer on the leftmost vertical line to the jth monomer on the rightmost vertical line. While this reconfiguration is taking place, the leftmost and rightmost vertical synchronization rows synchronize and delete themselves, leaving appropriate gadgets to connect the horizontal teeth: this signals the beginning of the next two half-doubling steps.

Expected Time, Space and States Analysis. Lemma 5 states that the expected time to perform a half-doubling is $O(\log m)$ for an $m \times m$ comb, and since $n \leq m$, the slowest half-doubling takes expected time $O(\log n)$. Each doubling involves 2 horizontal half-doubling phases, and 2 vertical half-doubling phases, and the 4 phases are separated by discrete synchronization events. Reconfiguration involves $O(n^2)$ bond and state change events, that take place independently and in parallel ($O(\log n)$ expected time) as well as a constant number of synchronizations that each take $O(\log n)$ expected time. Hence for $4(\lfloor \log_2 n \rfloor + 1)$ such half-doublings, plus $\lfloor \log_2 n \rfloor + 1$ reconfigurations, we get an overall expected time of $O(\log^2 n)$.

We've sketched how to make an $n \times n$ structure in $(n+2) \times (n+2)$ space. To make the construction work in $n \times n$ space, we first subtract 2 from the input, and build an $(n-2) \times (n-2)$ structure, and then at the final step have the leftmost and rightmost horizontal, and topmost and bottommost vertical, synchronization rows become rigid and be the border of the final $n \times n$ structure. A final monomer is added on the top left corner and we are done. By stepping through the construction it can be seen that $O(1)$ monomer states are sufficient. □

Intuitively, the following lemma holds because the long (length m) teeth allow for rapid, $O(1)$ time per tooth, and parallel insertion of monomers to expand the width of the comb. This intuition is complicated by the fact that teeth agitating open and closed may temporarily block other teeth inserting a new monomer. However, after an insertion actually happens further growth occurs independently and in parallel, taking logarithmic expected time overall.

Lemma 5. *A comb with m teeth where each tooth is of height m, can be horizontally half-doubled to length $\lfloor 1.5m \rfloor$ in expected time $O(\log m)$ using agitation nubots.*

Proof. Consider tooth i, where $1 \leq i \leq m$ for i even. A tooth can be `open`, `closed` or `initially filled` (one new monomer inserted). Although the remaining structure can affect the transition probabilities relevant to tooth i, in any state, the rate at which the tooth transitions from `closed` to `open` is at least m, the rate that it transitions from `open` to `closed` is at most m^2, and the rate at which it transitions from `open` to `initially filled` is exactly m. We define a new Markov process, with states `open`, `closed`, and `initially filled` and the transition probabilities just described, which is easier to analyze. Clearly, the random variable representing the time for this process to transition from `closed` to `initially filled` upper bounds the random variable representing the time for the real nubots process to do the same for a single tooth. We now show that this new random variable has expected value $O(1)$.

Let T_{cf} be the random variable representing the time to go from `closed` to `initially filled`. Let T_{co} be the random variable representing the time to go from `closed` to `open`. Let T_{oc} be the random variable representing the time to go from `open` to `closed`, conditioned on that transition happening, and define T_{of} similarly for going from `open` to `initially filled`. Note that $\mathrm{E}[T_{\mathrm{co}}] \leq \frac{1}{m}$, $\mathrm{E}[T_{\mathrm{oc}}] \geq \frac{1}{m^2}$, and $\mathrm{E}[T_{\mathrm{of}}] = \frac{1}{m}$. Let E_i represent the event that the process revisits state `closed` exactly i times after being in state `open` (and before reaching state `initially filled`). Let T_i be the random variable representing the time to take exactly i cycles between the states `open` and `closed`. Let C be the random variable representing the number of cycles taken between the states `open` and `closed` before transitioning to state `initially filled`. $\mathrm{E}[C] = m$ since the process goes from `open` to `initially filled` with probability $\frac{1}{m}$. Then

$$\begin{aligned}
\mathrm{E}[T_{\mathrm{cf}}] &= \mathrm{E}[T_{\mathrm{co}}] + \mathrm{E}[T_{\mathrm{of}}] + \sum_{i=0}^{\infty} \Pr[E_i] \cdot \mathrm{E}[T_i] \\
&\leq \frac{2}{m} + \sum_{i=0}^{\infty} \Pr[E_i] \cdot i \cdot \left(\frac{1}{m^2} + \frac{1}{m}\right) \\
&= \frac{2}{m} + \left(\frac{1}{m^2} + \frac{1}{m}\right) \sum_{i=0}^{\infty} \Pr[E_i] \cdot i \\
&= \frac{2}{m} + \left(\frac{1}{m^2} + \frac{1}{m}\right) \mathrm{E}[C] \\
&= \frac{2}{m} + \left(\frac{1}{m^2} + \frac{1}{m}\right) m \leq 2.
\end{aligned}$$

By Markov's inequality, the probability is at most $\frac{1}{2}$ that it will take more than time 4 to reach from `closed` to `initially filled`. Because of the memoryless property of the Markov process, conditioned on the fact that time t has elapsed without reaching state `initially filled`, the probability is at most $\frac{1}{2}$ that it will take more than $t + 4$ time to reach state `initially filled`. Hence for any $t > 0$, the probability that it will take more than than $4t$ time to reach from state `closed` to `initially filled` is at most 2^{-t}.

Since this tail probability decreases exponentially, it follows that for $m/2$ teeth, the expected time for all of them to reach state `initially filled` is $O(\log m)$. □

5 Building Lines via Agitation

In this section we build a line in sublinear time while using merely $O(n)$ space. We prove this, our second main theorem (Theorem 2), by giving a line construction that works in merely $n \times 5 = O(n)$ space while achieving sublinear expected time $O(n^{\epsilon+1/3})$, and $O(1)$ monomer states.

Proof (Theorem 2). **Overview of Construction.** The binary expansion of $n \in \mathbb{N}$ is encoded as a horizontal line, denoted x, of $\ell = \lfloor \log_2 n \rfloor + 1$ adjacent binary

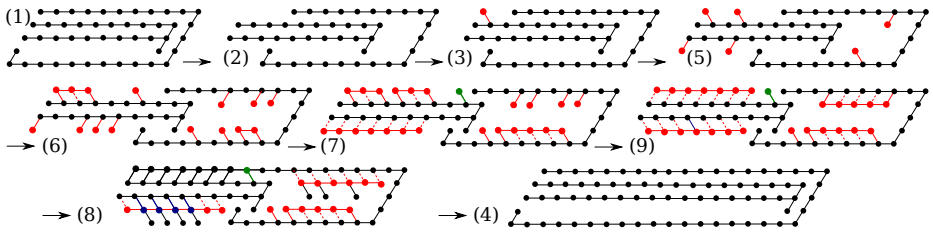

Fig. 7. Line doubling construction. The inner component is called the *sword*, which agitates left/right relative to the outer component called the scabbard (both are in black). The black sword-and-scabbard are doubled from length $m = 8$ to length $2m = 16$. Other monomers (red, green, blue) serve to both ratchet the movement, and to quickly in parallel build up the mass of the doubled sword-scabbard.

nubot monomers (each in state 0 or 1) with neighbouring monomers bound by rigid bonds, placed somewhere on the hexagonal grid. The leftmost of these monomers triggers the growth of a constant sized (length 1) sword and scabbard structure. Then an iterated doubling process begins, that happens exactly ℓ times and will result in a sword-and-scabbard of length n (and height 5). At step i of doubling, $1 \leq i \leq \ell$, the leftmost of the input monomers x_i (from x) is read, and then deleted. If $x_i = 0$ then there will be a doubling of the length of the sword-and-scabbard, else if $x_i = 1$ there will be a doubling of the length of the sword-and-scabbard with the addition of one extra monomer. It is straightforward to check that this doubling algorithm finishes with a length n object after ℓ rounds. After the final doubling step, a synchronization occurs, and then $\leq 4n$ of the monomers are deleted (in parallel) in such a way that an $n \times 1$ line remains. All that remains is to show the details of how each doubling step works.

Doubling Construction. Figure 7 describes the doubling process in detail: at iteration i of doubling assume that (a) we read an input bit 0, and that (b) we have a sword-and-scabbard structure of length m (and height 5). Since the input bit is 0 we want to double the length to $2m$. As shown in Figure 7(1), we begin with the sword sheathed in the scabbard. We next describe a biased (or ratcheted) random walk process that will ultimately result in the sword being withdrawn all the way to the *hook*, giving a structure of length $2m$. Via agitation, the sword may be unsheathed by moving out (to the left) of the scabbard, or by the scabbard moving (to the right) from the sword, although, because of the hook the sword can never be completely withdrawn and hence the two components will never drift apart.[1] The withdrawing of the sword is a random walk process with both the sword and scabbard agitating left-right. While this is happening, each monomer—at unit rate, conditioned on that monomer being unsheathed—on the top row of the sword tries to attach a new monomer above. Any such attachment event that succeeds acts as a *ratchet* that biases the ran-

[1] Besides preserving correctness of the construction, the hook is a safety feature, and hence the sword is merely decorative.

dom walk process in the forward direction. Also, as the sword is unsheathed each unsheathed sword monomer at the bottom of the sword attaches—at unit rate, conditioned on that monomer being unsheathed—a monomer below, and each monomer on the top (respectively, bottom) horizontal row of the scabbard tries to attach a monomer below (respectively, above) it. These monomers can also serve as ratchets (although in our time analysis below we ignore them which serves only to slow down the analysis). Eventually the sword is completely withdrawn to the hook, and ratcheted at that position, so further agitations do not change the structure.

At this point we are done with the doubling step, and the sword and scabbard reconfigure themselves to prepare for the next doubling (or deletion of monomers if we are done). Figure 7(6)–(9) gives the details. The attachment of new monomers results in 4 new horizontal line segments, each of length $m-1$. Each segment is built in the same way as used for the synchronization technique shown in Section 3, Figure 4; specifically the bonds are initially formed as rigid, and then transition to flexible in such a way that the line segment (or "synchronization" row) is free to agitate relative to its "backbone" row only when exactly all m bonds have formed. The line agitates left and right and is then synchronized (or locked into place, see Figure 4) causing all m monomers on the line to change state to "done". When the two new line segments that attached to the bottom and top of the sword are both done their rightmost monomers each bind to the scabbard with a rigid bond (as shown in Figure 7(8)) and delete their bonds to the sword (Figure 7(9)) (note that the rightmost of the latter kind of bonds is not deleted until after binding to the scabbard which ensures the entire structure remains connected at all times; also before the leftmost bond on the bottom is deleted a new hook is formed which prevents the new sword leaving the new scabbard prematurely). In a similar process, the two new line segments that are attached to the scabbard form a new hook, bind themselves to the sword, and then release themselves from the scabbard. We are new ready for the next stage of doubling.

The previous description assumed that the input bit is 0. If the input bit is instead 1 then after doubling both the sword and scabbard are increased in length by 1 monomer (immediately before forming the hook on the new scabbard).

After the final doubling stage then $O(n)$ monomers need to be deleted to leave an $n \times 1$ line of rigidly bound monomers (the goal is to build a line) without having monomers drift away (so as not to violate the space bound). This is relatively straightforward to achieve with two synchronizations, and subsequent deletion of monomers.

Expected Time Analysis. Lemma 6 states the expected time for a single doubling event: a length m sword is fully withdrawn to the hook, and locked into place, from a length m scabbard in expected time $O(m^{1/3+\epsilon})$.

Between each doubling event there is a reconfiguration of the sword and scabbard. Each reconfiguration invokes a constant number of synchronizations which, via Lemma 4, take expected time $O(\log m)$ each. Changing of the bond structure also takes place in $O(\log m)$ expected time since each of the four new line

segments change their bonds independently, and within a line segment all bond changes (expect for a constant number) occur independently and in parallel.

There are $\ell = \lfloor \log_2 n \rfloor + 1$ doubling plus reconfiguration events, each taking time $c((2^k)^{1/3+\epsilon})$ on the k'th event for some constant c (by Lemma 6, since the size of the structure during the k'th event is $\Theta(2^k)$), the total expected time is bounded by the geometric series

$$\sum_{k=0}^{\ell-1} c(2^k)^{1/3+\epsilon} = c\sum_{k=0}^{\ell-1}(2^{1/3+\epsilon})^k = c\frac{1-(2^{1/3+\epsilon})^\ell}{1-2^{1/3+\epsilon}} = O((2^{1/3+\epsilon})^\ell) = O(n^{1/3+\epsilon}).$$

□

The next lemma states that, starting from length m, one "length-doubling" stage of the 1D line construction completes in expected time $O(m^{1/3+\epsilon})$. Intuitively, the proof (see full paper) shows that the rapid agitation process is a random walk that quickly exposes a large portion of the sword, to which a monomer quickly attaches. This attachment irreversibly "ratchets" the random walk forward, preventing it from walking backwards by very much.

Lemma 6. *For any $\epsilon > 0$, for sufficiently large m, the expected time for one line-doubling stage (doubling the length of the sword and scabbard) is $O(m^{1/3+\epsilon})$.*

Acknowledgments. A special thanks to Erik Winfree for many insightful and helpful discussions on the model and constructions. We also thank Robert Schweller, Matthew Cook and Andrew Winslow for discussions on the model and problems studied in this paper.

References

1. Bath, J., Turberfield, A.: DNA nanomachines. Nature Nanotechnology 2, 275–284 (2007)
2. Chen, M., Xin, D., Woods, D.: Parallel computation using active self-assembly. In: Soloveichik, D., Yurke, B. (eds.) DNA 2013. LNCS, vol. 8141, pp. 16–30. Springer, Heidelberg (2013); Full version: `arXiv:1405.0527`
3. Dabby, N., Chen, H.-L.: Active self-assembly of simple units using an insertion primitive. In: SODA: Proceedings of the Twenty-fourth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1526–1536 (January 2012)
4. Dawes-Hoang, R.E., Parmar, K.M., Christiansen, A.E., Phelps, C.B., Brand, A.H., Wieschaus, E.F.: Folded gastrulation, cell shape change and the control of myosin localization. Development 132(18), 4165–4178 (2005)
5. Doty, D.: Theory of algorithmic self-assembly. Communications of the ACM 55, 78–88 (2012)
6. Doty, D., Lutz, J.H., Patitz, M.J., Schweller, R.T., Summers, S.M., Woods, D.: The tile assembly model is intrinsically universal. In: FOCS: Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science, pp. 439–446 (October 2012)

7. Martin, A.C., Kaschube, M., Wieschaus, E.F.: Pulsed contractions of an actin–myosin network drive apical constriction. Nature 457(7228), 495–499 (2008)
8. Patitz, M.J.: An introduction to tile-based self-assembly. In: Durand-Lose, J., Jonoska, N. (eds.) UCNC 2012. LNCS, vol. 7445, pp. 34–62. Springer, Heidelberg (2012)
9. Winfree, E.: Algorithmic Self-Assembly of DNA. PhD thesis, California Institute of Technology (June 1998)
10. Woods, D.: Intrinsic universality and the computational power of self-assembly. In: MCU: Proceedings of Machines, Computations and Universality, September 9-12. Electronic Proceedings in Theoretical Computer Science, vol. 128, pp. 16–22 (2013)
11. Woods, D., Chen, H.-L., Goodfriend, S., Dabby, N., Winfree, E., Yin, P.: Active self-assembly of algorithmic shapes and patterns in polylogarithmic time. In ITCS 2013: Proceedings of the 4th conference on Innovations in Theoretical Computer Science, pp. 353–354. ACM (2013) Full version: `arXiv:1301.2626` [cs.DS]