

The Power of Nondeterminism in Self-Assembly*

Nathaniel Bryans[†] Ehsan Chiniforooshan[†] David Doty[‡] Lila Kari[†]
Shinnosuke Seki[†]

Abstract

We investigate the role of nondeterminism in Winfree’s abstract tile assembly model, which was conceived to model artificial molecular self-assembling systems constructed from DNA. By nondeterminism we do not mean a magical ability such as that possessed by a nondeterministic algorithm to search an exponential-size space in polynomial time. Rather, we study realistically implementable systems that retain a different sense of determinism in that they are guaranteed to produce a unique shape but are nondeterministic in that they do not guarantee which tile types will be placed where within the shape.

We show a “molecular computability” result: there is an infinite shape S that is uniquely assembled by a tile system but not by any deterministic tile system. We show a “molecular complexity” result: there is a finite shape S that is uniquely assembled by a tile system with c tile types, but every deterministic tile system that uniquely assembles S has more than c tile types. In fact we extend the technique to derive a stronger (classical complexity theoretic) result, showing that the problem of finding the minimum number of tile types that uniquely assemble a given finite shape is Σ_2^P -complete. In contrast, the problem of finding the minimum number of *deterministic* tile types that uniquely assemble a shape is NP-complete [5].

1 Introduction

Tile self-assembly is an algorithmically rich model of “programmable crystal growth”. It is possible to design molecules (square-like “tiles”) with specific binding sites so that, even subject to the chaotic nature of molecules floating randomly in a well-mixed chemical soup, they are guaranteed to bind so as to deterministically form a single target shape. This is despite the number of different types of tiles possibly being much smaller than the size of the shape and therefore having only “local information” to guide their attachment. The ability to control nanoscale structures and machines to atomic-level precision will rely crucially on sophisticated

self-assembling systems that automatically control their own behavior where no top-down externally controlled device could fit.

A practical implementation of self-assembling molecular tiles was proved experimentally feasible in 1982 by Seeman [38] using DNA complexes formed from artificially synthesized strands. Experimental advances have delivered increasingly reliable assembly of algorithmic DNA tiles with error rates of 10% per tile in 2004 [36], 1.4% per tile in 2007 [17], and 0.13% per tile in 2009 [8]. Erik Winfree [44] introduced the abstract Tile Assembly Model (aTAM) – based on a constructive version of Wang tiling [42, 43] – as a simplified mathematical model of self-assembling DNA tiles. Winfree 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 [35] investigated a self-assembly resource bound known as *tile complexity*, the minimum number of tile types needed to assemble a shape. They showed that for most n , the problem of assembling an $n \times n$ square has tile complexity $\Omega\left(\frac{\log n}{\log \log n}\right)$, and Adleman, Cheng, Goel, and Huang [4] exhibited a construction showing that this lower bound is asymptotically tight. Under natural generalizations of the model [2, 6, 9–13, 19, 20, 29, 39, 41], tile complexity can be reduced for tasks such as square-building and assembly of more general shapes.

There are different interpretations of “nondeterminism” in the aTAM. We say a tile system is *directed* (*a.k.a. deterministic*) if it is guaranteed to form one unique final assembly, where an assembly is defined not only by which positions are eventually occupied by a tile, but also by which tile type is placed at each position. We say a tile system *strictly* (*a.k.a. uniquely self-assembles*) a shape if all of its final assemblies are guaranteed to have that shape. A natural analogy may be made between a non-directed tile system that strictly self-assembles some shape and a nondeterministic Turing machine N that always produces the same output on a given input, regardless of the nondeterministic choices made during computation. There is always a deterministic Turing machine M computing the same function

*Supported by NSERC Discovery Grant R2824A01 and the Canada Research Chair Award in Biocomputing to Lila Kari, by the NSERC Undergraduate Student Research Awards (USRA) grant to Nathaniel Bryans, and by the NSF Computing Innovation Fellowship grant to David Doty.

[†]University of Western Ontario, Dept. of Computer Science, London, Ontario, Canada, N6A 5B7, {nbryans, ehsan, lila, sseki}@csd.uwo.ca

[‡]California Institute of Technology, Dept. of Computer Science, Pasadena, CA 91125, USA, ddoty@caltech.edu

as N and using no more “resources”, according to any common resource bound such as time complexity, space complexity, or program length. Therefore we regard such a restricted class of nondeterministic Turing machines as no more “powerful” than deterministic Turing machines.

Based on this analogy, it might seem that strict self-assembly, while allowing one form of nondeterminism (which tile goes where), so strongly requires another form of determinism (which positions have a tile) that extra power cannot be gained by allowing the tile systems to be non-directed. More precisely, it is natural to conjecture that every infinite shape that is strictly self-assembled by some tile system is also strictly self-assembled by some directed tile system. In the finitary case, *every* finite shape is assembled by a directed tile system (possibly using as many tile types as there are points in the shape), so to make the idea non-trivial we might conjecture that the tile complexity of a finite shape is independent of whether we consider all tile systems or only those that are directed. Such conjectures are appealing because the algorithmic design and verification of tile systems [39] as well as lower bounds and impossibility proofs [6, 15, 30] often rely on reasoning about directed tile systems, which are “better behaved” in many senses than arbitrary tile systems, even those that strictly self-assemble a shape. It would be helpful to begin such arguments with the phrase, “Assume without loss of generality that the tile system is directed.”

However, these conjectures are false. We show that there is an infinite shape S that is strictly self-assembled by a tile system but not by any directed tile system. Therefore, in a “molecular computability theoretic” sense, nondeterminism allows certain shapes to be algorithmically self-assembled that are totally “unassemblable” (to borrow Adleman’s tongue-twisting analog of “uncomputable” [3]) under the constraint of determinism. We then show an analogous phenomenon in the finitary case: there is a finite shape S that is strictly self-assembled by a tile system with c tile types, but every directed tile system that strictly self-assembles S has more than c tile types. In fact to strictly self-assemble S in a directed tile system requires at least $\approx \frac{3}{2}c$ tile types. It is open to prove a super-linear gap between the complexity measures; the issue is discussed in more detail in Section 5. This establishes a “molecular complexity theoretic” analog of the first result. We then derive a stronger result, showing that the problem of finding the minimum number of tile types that strictly self-assemble a given finite shape is complete for the complexity class $\Sigma_2^P = \text{NP}^{\text{NP}}$. In contrast, the problem of finding the minimum number of

directed tile types that strictly self-assemble a shape was shown to be NP-complete by Adleman, Cheng, Goel, Huang, Kempe, Moisset de Espanés, and Rothmund [5].

Based on these results, we conclude that nondeterminism confers extra power to assemble a shape from a small tile system, but unless the polynomial hierarchy collapses, it is computationally more difficult¹ to exploit this power by finding the size of the smallest tile system, compared to finding the size of the smallest directed tile system.

One might argue that this difference between nondeterministic (but “output-deterministic”) Turing machines and non-directed (but strict) tile systems is not surprising, since there is a “monotone” aspect to tile assembly in the sense that space used to place a tile cannot be reused, whereas a tape cell used to store information by a Turing machine can be reused to store different information later. It is sometimes said that the difference between space and time is that “you can reuse space but you cannot reuse time.” However, the “computation” carried out by tile systems does not distinguish well between space and time. For instance, the standard simulation of a Turing machine by a tile system (see [35]) assembles a structure encoding the entire space-time configuration history of the Turing machine. Even with negative glue strengths that are able to force detachments to occur, the volume requirements of such a simulation must be proportional to $t \cdot s$ for a Turing machine using time t and space s [14], essentially forcing the solution to contain multiple assemblies that collectively encode the entire computation history. Tile systems therefore cannot reuse space (tiles), which is the fundamental effect of their monotonicity on their computational abilities. From this perspective, a Turing machine cannot reuse time any better than any other computational system (barring the use of closed timelike curves [1]), including tile systems. Yet in contrast to tile systems, a nondeterministic but “output-deterministic” Turing machine remains no more powerful, even in the sense of time complexity, than a deterministic Turing machine.

2 Abstract Tile Assembly Model

This section gives a terse definition of the abstract Tile Assembly Model (aTAM, [44]). This is not a tutorial; for readers unfamiliar with the aTAM, [35] gives an excellent introduction to the model.

Fix an alphabet Σ . Σ^* is the set of finite strings over

¹“More difficult” in the sense of nondeterministic time complexity, although it is conceivable that both problems have the same deterministic time complexity.

Σ . Given a discrete object O , $\langle O \rangle$ denotes a standard encoding of O as an element of Σ^* . \mathbb{Z} , \mathbb{Z}^+ , and \mathbb{N} denote the set of integers, positive integers, and nonnegative integers, respectively. For a set A , $\mathcal{P}(A)$ denotes the power set of A . Given $A \subseteq \mathbb{Z}^2$, the *full grid graph* of A is the undirected graph $G_A^f = (V, E)$, where $V = A$, and for all $u, v \in V$, $\{u, v\} \in E \iff \|u - v\|_2 = 1$; i.e., iff u and v are adjacent on the integer Cartesian plane. A *shape* is a set $S \subseteq \mathbb{Z}^2$ such that G_S^f is connected. A shape Υ is a *tree* if G_Υ^f is acyclic.

A *tile type* is a tuple $t \in (\Sigma^* \times \mathbb{N})^4$; i.e., a unit square with four sides listed in some standardized order, each side having a *glue* $g \in \Sigma^* \times \mathbb{N}$ consisting of a finite string *label* and 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 nonempty connected arrangement of tiles on the integer lattice \mathbb{Z}^2 , i.e., a partial function $\alpha : \mathbb{Z}^2 \dashrightarrow T$ such that $G_{\text{dom } \alpha}^f$ is connected and $\text{dom } \alpha \neq \emptyset$. The *shape* $S_\alpha \subseteq \mathbb{Z}^2$ of α is $\text{dom } \alpha$. Two adjacent tiles in an assembly *interact* if the glues on their abutting sides are equal (in both label and strength) and have positive strength. Each assembly α induces a *binding graph* G_α^b , a grid graph whose vertices are positions occupied by tiles, with an edge between two vertices if the tiles at those vertices interact.² Given $\tau \in \mathbb{Z}^+$, α is τ -*stable* if every cut of G_α^b has weight at least τ , where the weight of an edge is the strength of the glue it represents. That is, α is τ -stable if at least energy τ is required to separate α into two parts. When τ is clear from context, we say α is *stable*. Given two assemblies $\alpha, \beta : \mathbb{Z}^2 \dashrightarrow T$, we say α is a *subassembly* of β , and we write $\alpha \sqsubseteq \beta$, if $S_\alpha \subseteq S_\beta$ and, for all points $p \in S_\alpha$, $\alpha(p) = \beta(p)$.

A *tile assembly system* (TAS) is a triple $\mathcal{T} = (T, \sigma, \tau)$, where T is a finite set of tile types, $\sigma : \mathbb{Z}^2 \dashrightarrow T$ is the finite, τ -stable *seed assembly*, and $\tau \in \mathbb{Z}^+$ is the *temperature*. Given two τ -stable assemblies $\alpha, \beta : \mathbb{Z}^2 \dashrightarrow T$, we write $\alpha \rightarrow_1^\mathcal{T} \beta$ if $\alpha \sqsubseteq \beta$ and $|S_\beta \setminus S_\alpha| = 1$. In this case we say α \mathcal{T} -*produces* β *in one step*.³ If $\alpha \rightarrow_1^\mathcal{T} \beta$, $S_\beta \setminus S_\alpha = \{p\}$, and $t = \beta(p)$, we write $\beta = \alpha + (p \mapsto t)$. The \mathcal{T} -*frontier* of α is the set $\partial^\mathcal{T} \alpha = \bigcup_{\alpha \rightarrow_1^\mathcal{T} \beta} S_\beta \setminus S_\alpha$, the set of empty locations at which a tile could stably attach to α .

²For $G_{S_\alpha}^f = (V_{S_\alpha}, E_{S_\alpha})$ and $G_\alpha^b = (V_\alpha, E_\alpha)$, G_α^b is a spanning subgraph of $G_{S_\alpha}^f$: $V_\alpha = V_{S_\alpha}$ and $E_\alpha \subseteq E_{S_\alpha}$.

³Intuitively $\alpha \rightarrow_1^\mathcal{T} \beta$ means that α can grow into β by the addition of a single tile; the fact that we require both α and β to be τ -stable implies in particular that the new tile is able to bind to α with strength at least τ . It is easy to check that had we instead required only α to be τ -stable, and required that the cut of β separating α from the new tile has strength at least τ , then this implies that β is also τ -stable.

A sequence of $k \in \mathbb{Z}^+ \cup \{\infty\}$ assemblies $\alpha_0, \alpha_1, \dots$ is a \mathcal{T} -*assembly sequence* if, for all $1 \leq i < k$, $\alpha_{i-1} \rightarrow_1^\mathcal{T} \alpha_i$. We write $\alpha \rightarrow^\mathcal{T} \beta$, and we say α \mathcal{T} -*produces* β (in 0 or more steps) if there is a \mathcal{T} -assembly sequence $\alpha_0, \alpha_1, \dots$ of length $k = |S_\beta \setminus S_\alpha| + 1$ such that 1) $\alpha = \alpha_0$, 2) $S_\beta = \bigcup_{0 \leq i < k} S_{\alpha_i}$, and 3) for all $0 \leq i < k$, $\alpha_i \sqsubseteq \beta$. If k is finite then it is routine to verify that $\beta = \alpha_{k-1}$.⁴ We say α is \mathcal{T} -*producible* if $\sigma \rightarrow^\mathcal{T} \alpha$, and we write $\mathcal{A}[\mathcal{T}]$ to denote the set of \mathcal{T} -producible assemblies. The relation $\rightarrow^\mathcal{T}$ is a partial order on $\mathcal{A}[\mathcal{T}]$ [23, 34]. A \mathcal{T} -assembly sequence $\alpha_0, \alpha_1, \dots$ is *fair* if, for all i and all $p \in \partial^\mathcal{T} \alpha_i$, there exists j such that $\alpha_j(p)$ is defined; i.e., no frontier location is “starved”.

An assembly α is \mathcal{T} -*terminal* if α is τ -stable and $\partial^\mathcal{T} \alpha = \emptyset$. We write $\mathcal{A}_\square[\mathcal{T}] \subseteq \mathcal{A}[\mathcal{T}]$ to denote the set of \mathcal{T} -producible, \mathcal{T} -terminal assemblies. A TAS \mathcal{T} is *directed* (a.k.a., *deterministic*, *confluent*) if the poset $(\mathcal{A}[\mathcal{T}], \rightarrow^\mathcal{T})$ is directed; i.e., if for each $\alpha, \beta \in \mathcal{A}[\mathcal{T}]$, there exists $\gamma \in \mathcal{A}[\mathcal{T}]$ such that $\alpha \rightarrow^\mathcal{T} \gamma$ and $\beta \rightarrow^\mathcal{T} \gamma$.⁵ We say that a TAS \mathcal{T} *strictly* (a.k.a. *uniquely*) *self-assembles* a shape $S \subseteq \mathbb{Z}^2$ if, for all $\alpha \in \mathcal{A}_\square[\mathcal{T}]$, $S_\alpha = S$; i.e., if every terminal assembly produced by \mathcal{T} has shape S . If \mathcal{T} strictly self-assembles some shape S , we say that \mathcal{T} is *strict*. Note that the implication “ \mathcal{T} is directed $\implies \mathcal{T}$ is strict” holds, but the converse does not hold.

In this paper we will always use *singly-seeded temperature-2* TAS’s, those with $|S_\sigma| = 1$ and $\tau = 2$; hence we will use the term *seed tile* for σ as well, and for the remainder of this paper we use the term TAS to mean singly-seeded temperature-2 TAS. When \mathcal{T} is clear from context, we may omit \mathcal{T} from the notation above and instead write $\rightarrow_1, \rightarrow, \partial\alpha, \text{frontier}, \text{assembly sequence}, \text{produces}, \text{producible}, \text{and terminal}$. Since the behavior of a TAS $\mathcal{T} = (T, \sigma, 2)$ is unchanged if every glue with strength greater than 2 is changed to have strength exactly 2, we assume henceforth that all glue strengths are 0, 1, or 2, and use the terms *null glue*, *single glue*, and *double glue*, respectively, to refer to these three cases.⁶ We also assume without loss of generality that every single glue or double glue occurring in some tile type in some direction also occurs in some tile type in the opposite direction, i.e., there are no “effectively

⁴If we had defined the relation $\rightarrow^\mathcal{T}$ based on only finite assembly sequences, then $\rightarrow^\mathcal{T}$ would be simply the reflexive, transitive closure $(\rightarrow_1^\mathcal{T})^*$ of $\rightarrow_1^\mathcal{T}$. But this would mean that no infinite assembly could be produced from a finite assembly, even though there is a well-defined, unique “limit assembly” of every infinite assembly sequence.

⁵The following two convenient characterizations of “directed” are routine to verify. \mathcal{T} is directed if and only if $|\mathcal{A}_\square[\mathcal{T}]| = 1$. \mathcal{T} is not directed if and only if there exist $\alpha, \beta \in \mathcal{A}[\mathcal{T}]$ and $p \in S_\alpha \cap S_\beta$ such that $\alpha(p) \neq \beta(p)$.

⁶We use *null bond*, *single bond*, and *double bond* similarly to refer to the *interaction* of two tiles.

null” single or double glues.⁷

3 Assembly of Infinite Shapes

In this section we study the power of nondeterminism in assembling infinite shapes. The following theorem is the main result of Section 3.

THEOREM 3.1. *There is a shape $S \subset \mathbb{Z}^2$ such that some TAS strictly self-assembles S , but no directed TAS strictly self-assembles S .*

Proof. Let $L \subset \mathbb{N}$ be a language that is computably enumerable but not decidable, and let M be a Turing machine such that $L = L(M)$. Let S be the shape that is strictly self-assembled by the TAS described below, when M is encoded into the TAS as described.

A portion of the shape S is shown in Figure 1. The TAS that strictly self-assembles S is based on the main construction of Lathrop, Lutz, Patitz, and Summers [22]. In that paper, the authors show that for each Turing machine M , an encoding of the language $L(M) \subseteq \mathbb{N}$ accepted by M “weakly self-assembles” on the x -axis. More precisely, for a “reasonably simple” function $f : \mathbb{N} \rightarrow \mathbb{N}$, a special tile type is placed at position $(f(n), 0)$ if and only if $n \in L(M)$. The n^{th} “ray” in Figure 1 begins growth just before $(f(n), 0)$, and grows independently of the other rays, controlling an adjacent simulation of $M(n)$ in parallel with all the other rays. The slope of each ray is just a bit smaller than the previous, with the slope approaching 2 as $n \rightarrow \infty$. The simulation executes one transition of M on input n every $\approx 2^n$ rows of the ray. Since M can use no more than k tape cells after k transitions, this slowed simulation ensures that each ray has enough space to allow a potentially unbounded simulation of M on each n , without “crashing” into the next adjacent ray, even in the worst case that M moves its tape head right on every transition.

What is needed from this construction for our purpose is:

1. f is computable.⁸

⁷Thus the existence of a tile with a double glue facing empty space implies that the empty space is part of the frontier. Many of our arguments use the contrapositive that if a shape S is strictly self-assembled by a tile system and a side of a tile faces a point $p \notin S$, then the tile cannot have a double glue on that side.

⁸[22] defines the roughly quadratic function $f(n) = \binom{n+1}{2} + (n+1) \lfloor \log n \rfloor + 6n - 2^{1+\lfloor \log n \rfloor} + 2$. Our version of this function will grow just a bit faster, to make room for a vertical line to form between two adjacent rays without “touching” the rest of the shape except at the endpoints of the line, but retains computability.

2. The simulation of $M(n)$, carried out adjacent to the n^{th} ray, sends a “signal” crawling down the right side of the simulation if and only if M accepts n , placing a special tile just above the “planter” (the group of tiles growing below each of the rays).

We modify the signal so that, rather than growing all the way to the planter, for input n , the signal grows to distance n north of the planter and then grows a width-1 vertical line n positions down to the planter, using the same tile type with equal north and south double glues to “crash” into the planter. To ensure that the downward-growing vertical lines do not obstruct the operation of the planter, the planter is modified so that it is guaranteed to grow horizontally a sufficient number of tiles before laying out the input for the M , so as to guarantee that there is something present for a “controlled crash.” The space for the downward-growing line of length n to drop after the input is accepted is created by having the Turing machine simulations begin not immediately above the planter, but at height n on input n . This is why the n^{th} ray grows straight up for n rows before beginning its sloped growth. Under every simulation, a “notch” tile is placed above the planter using a double glue, which is horizontally lined up with where the vertical line will grow if M accepts. The actions of the ray, planter and Turing machine simulation are otherwise similar to the mechanisms used in [22]. We note that this particular TAS is not directed since the “notch” tiles compete nondeterministically with the vertical line tiles at positions where M accepts.

It remains to show that no directed TAS strictly self-assembles S . Intuitively, we show that at points of the form $(f(n), 0)$, any directed TAS must place tiles that “know” whether there will eventually be a vertical line above the point, implying the ability to decide L since vertical lines appear above exactly those positions $(f(n), 0)$ such that $n \in L$. Assume for the sake of contradiction that there is a directed TAS $\mathcal{T} = (T, \sigma, 2)$ that strictly self-assembles S , and let $\alpha \in \mathcal{A}_{\square}[\mathcal{T}]$ be its unique producible, terminal assembly. Since the heights of the vertical “bases” of each ray below the sloped portion are strictly increasing, there is some $n_0 \in \mathbb{N}$ such that, for all $n > n_0$, the distance from $(f(n), 1)$ to the ray above it is at least $|T| + 1$. Let $Y = \{ (f(n), 0) \mid n \in L \text{ and } n > n_0 \}$ be the bottommost points of the vertical lines adjacent to rays corresponding to (sufficiently large) “yes” instances of L , and let $N = \{ (f(n), 0) \mid n \notin L \text{ and } n > n_0 \}$ represent the positions of the “notches” corresponding to (sufficiently large) “no” instances. Y and N are shown in Figure 1. Let $T_Y = \alpha(Y)$ and $T_N = \alpha(N)$ be the set of tile types that appear at “yes” and “no”

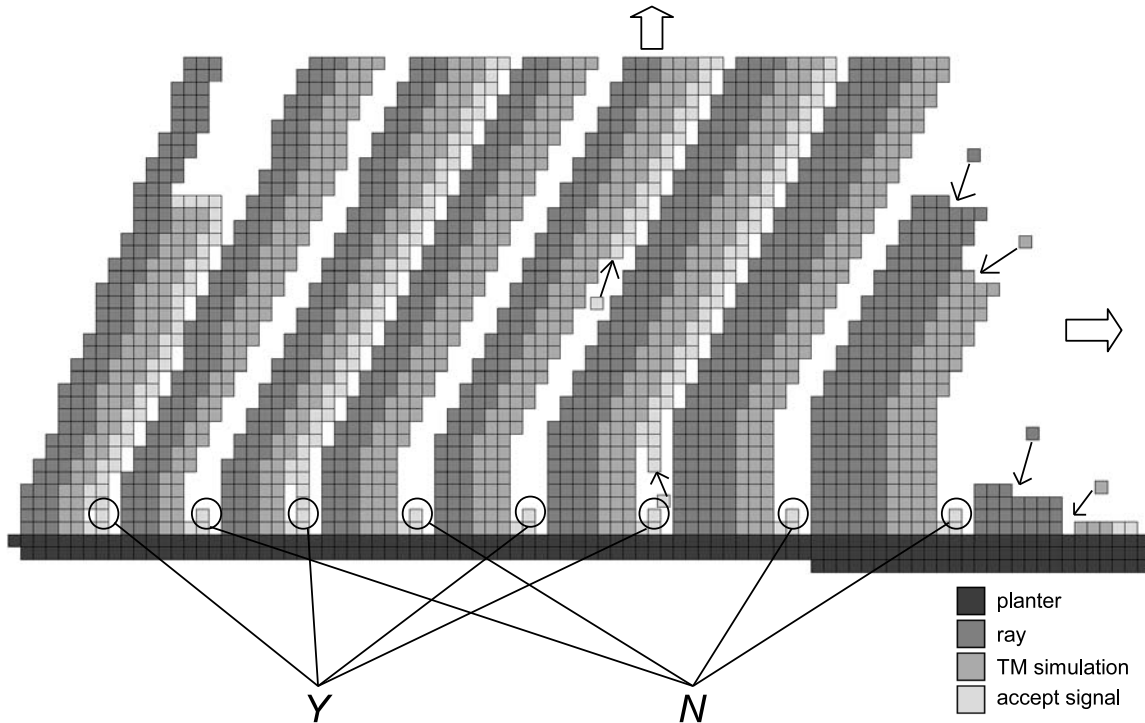


Figure 1: A portion of an infinite shape S that strictly self-assembles, but not by any directed TAS. The n^{th} ray simulates a Turing machine M on input n , and a vertical line is present under that ray if and only if M accepts n . Y and N are points at these positions representing “yes” and “no” instances of $L(M)$, respectively, and elements of Y are the points where nondeterminism is forced to occur in any TAS that strictly self-assembles S .

instance points, respectively. Since S has empty space immediately north of positions in N , no tile type in T_N has a north double glue.

We claim that $T_Y \cap T_N = \emptyset$. For the sake of contradiction, suppose otherwise, let $t \in T_Y \cap T_N$, and let $p \in Y$ be a point where $\alpha(p) = t$. Since $t \in T_N$, t has no north double glue, so the vertical line above p must grow downward using north and south double glues. Let $q = p + (0, 1)$ be the point just above p . By our choice of n_0 , the vertical line must repeat a tile type before reaching the point q , so all tile types in the repetition period have a north and a south double glue, including the tile type $t' = \alpha(q)$. Let t'' be the tile type appearing beneath t' after the previous occurrence of t' in the vertical line. Since t'' has a north double glue, $t'' \notin T_N$, so $t'' \neq t$. Because t binds to the rest of α only through its south double glue, there can be no precedence relationship enforcing that p must contain a tile before q (or any other point) receives a tile. In other words, there exists a producible assembly $\beta \in \mathcal{A}[\mathcal{T}]$ such that $\beta(q) = t'$ and $\beta(p)$ is undefined. This implies that t'' can bind to β at position p to create $\beta' = \beta + (p \mapsto t'')$,

contradicting the directedness of \mathcal{T} since $\beta', \alpha \in \mathcal{A}[\mathcal{T}]$ but $\beta'(p) = t'' \neq t = \alpha(p)$. This verifies the claim that $T_Y \cap T_N = \emptyset$.

For all $n \in \mathbb{N}$, let $p_n = (f(n), 0)$. Since $T_Y \cap T_N = \emptyset$, for all $n > n_0$, $n \in L \iff p_n \in Y \iff \alpha(p_n) \in T_Y$, and $n \notin L \iff p_n \in N \iff \alpha(p_n) \in T_N$. Using this fact, we describe an algorithm to decide L , contradicting its undecidability and completing the proof. On input $n \in \mathbb{N}$, if $n \leq n_0$, use a constant lookup table to decide n . Otherwise, compute $p_n = (f(n), 0)$. Simulate the assembly of \mathcal{T} with a fair assembly sequence, maintaining a first-in, first-out queue of frontier locations to enforce fairness, until a tile is placed at position p_n . Since this assembly sequence is fair, the simulation will eventually place a tile type $\alpha(p_n)$ at p_n , and $\alpha(p_n)$'s membership in T_Y or T_N will indicate whether to accept or reject n . \square

We have implemented the tile assembly system that strictly self-assembles S :

<http://www.dna.caltech.edu/~ddoty/pnsa/>

It can be simulated using Matthew Patitz's ISU TAS simulator [33]:

<http://www.cs.iastate.edu/~lnsa/software.html>

The purpose of the implementation is not to quantitatively analyze the construction, since we make no quantitative claims about either the shape being assembled nor the TAS that strictly self-assembles the shape. Furthermore, the bulk of the intellectual effort in the proof of Theorem 3.1 is proving the negative result that no directed TAS strictly self-assembles the shape, which is something that cannot be established through a simulation. We provide the simulation primarily to help the interested reader understand the details of the construction.

4 Assembly of Finite Shapes

In this section we study the power of nondeterminism in assembling finite shapes. We first show that a finitary analog of Theorem 3.1 holds, by showing that the tile complexity of some shapes can be reduced using nondeterminism. The ideas in this construction will be useful in proving the main theorem of this section, which shows that the minimum tile set problem is Σ_2^P -complete.

Recall that all of the TAS's we study are assumed singly-seeded. Let $S \subseteq \mathbb{Z}^2$ be a shape. The (*temperature-2*) *tile complexity* of S is

$$C^{tc}(S) = \min \left\{ |T| \mid \begin{array}{l} \mathcal{T} = (T, \sigma, 2) \text{ is a} \\ \text{TAS and } \mathcal{T} \text{ strictly} \\ \text{self-assembles } S \end{array} \right\},$$

with $\min \emptyset = \infty$. The (*temperature-2*) *directed tile complexity* of S is

$$C^{dtc}(S) = \min \left\{ |T| \mid \begin{array}{l} \mathcal{T} = (T, \sigma, 2) \text{ is a directed} \\ \text{TAS and } \mathcal{T} \text{ strictly} \\ \text{self-assembles } S \end{array} \right\}.$$

We are interested in the problems, given a finite shape, what is its tile complexity, and what is its directed tile complexity? We define two decision problems that are equivalent to these optimization problems. Let $\mathcal{FS} \subset \mathcal{P}(\mathbb{Z}^2)$ denote the set of all finite shapes. The *minimum tile set* problem is

$$\text{MINTILESET} = \left\{ \langle S, c \rangle \mid \begin{array}{l} S \in \mathcal{FS}, c \in \mathbb{Z}^+, \\ \text{and } C^{tc}(S) \leq c \end{array} \right\},$$

and the *minimum directed tile set* problem is

$$\text{MINDIRTILESET} = \left\{ \langle S, c \rangle \mid \begin{array}{l} S \in \mathcal{FS}, c \in \mathbb{Z}^+, \\ \text{and } C^{dtc}(S) \leq c \end{array} \right\}.$$

Adleman, Cheng, Goel, Huang, Kempe, Moisset de Espanés, and Rothmund [5] showed that the problem MINDIRTILESET is NP-complete. In Section 4.2 we show that MINTILESET is Σ_2^P -complete, where $\Sigma_2^P = \text{NP}^{\text{NP}}$. See [7] for a discussion of these complexity classes.

4.1 A Finite Shape for which Nondeterminism Reduces Tile Complexity.

Although the main result of Section 4, Theorem 4.2, together with the (widely-believed) assumption that $\text{NP} \neq \Sigma_2^P$ and the fact proven in [5] that $\text{MINDIRTILESET} \in \text{NP}$, implies Theorem 4.1 of this subsection, we define the shape of Theorem 4.1 explicitly in order to illustrate some of the reasoning used in the proof of Theorem 4.2.

THEOREM 4.1. *There is a finite shape $S \subset \mathbb{Z}^2$ such that $C^{tc}(S) < C^{dtc}(S)$.*

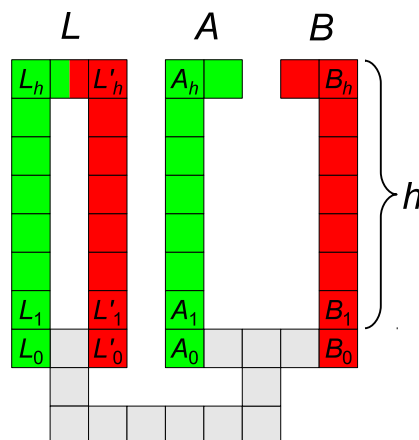


Figure 2: A finite shape S for which $C^{tc}(S) < C^{dtc}(S)$. Nondeterminism is forced to occur at the two-color top-middle position of the loop L , since any minimal tile set must reuse the tile types from subtrees A and B to create L .

The shape is shown in Figure 2. A formal proof of Theorem 4.1 is contained in the full version of this extended abstract. Intuitively, the tile complexity is about $2h$ since the tile types for the two “pillars” on the right can be reused to create the two more closely packed pillars on the left. Since the right half of the shape is a tree, all tile types in that region must be unique and therefore already contribute $2h$ to the tile complexity of the whole shape. In fact, any attempt to use fewer than $3h$ tile types necessarily introduces competition between tile types by forcing us to reuse tile types from the right two pillars to assemble the left two pillars.

4.2 The Minimum Tile Set Problem is Σ_2^P -complete.

The following is the main theorem of Section 4.

THEOREM 4.2. *MINTILESET is Σ_2^P -complete.*

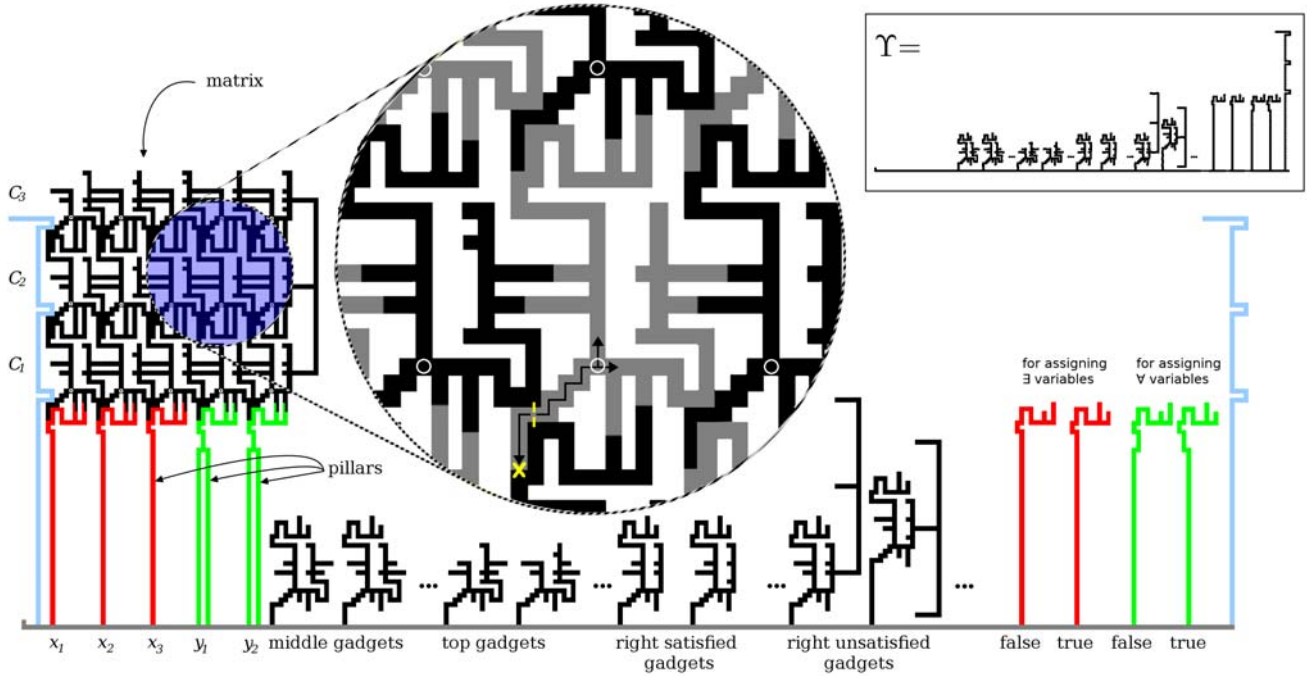


Figure 3: The shape S of the reduction $\langle \varphi \rangle \mapsto \langle S, c \rangle$ showing $\exists\forall\text{CNF-UNSAT} \leq_m^P \text{MINTILESET}$. In this example, the quantified negated CNF formula $\varphi = \exists x\forall y\neg\phi(x, y)$ has clauses C_1, C_2 and C_3 , \exists -variables x_1, x_2 , and x_3 , and \forall -variables y_1 and y_2 . The “matrix” of gadgets at the top left has a row of gadgets for each clause and a column of gadgets for each variable. The matrix sits atop a group of “pillars” that, when tiled by actual tiles, will represent a variable assignment to ϕ (along with one taller left-boundary pillar to help initiate cooperative binding of gadgets to assemble the matrix). The tree Υ is S without the matrix and pillars beneath it. In the zoom-in, the two yellow lines above the yellow X represent strength-1 glues that cooperate to place the gray gadget once (enough of) the black gadgets to its west and south are in place. The yellow X shows “backward growth” of the gray gadget that is blocked before it can grow down far enough to form a new copy of the bottom row of S .

Proof. (Sketch). To show that $\text{MINTILESET} \in \Sigma_2^P$, $\exists\forall\text{CNF-UNSAT}$ is the Σ_2^P -complete language [37, 40, 45] define the verification language

$$\text{MINTILESET}_V = \left\{ \langle S, c, \mathcal{T}, \vec{\alpha} \rangle \mid \begin{array}{l} S \in \mathcal{FS}, c \in \mathbb{Z}^+, \mathcal{T} = (T, \sigma, 2) \text{ is} \\ \text{a TAS with } |T| \leq c, \vec{\alpha} = \\ (\sigma, \alpha_2, \alpha_3, \dots, \alpha_k) \text{ is a} \\ \mathcal{T}\text{-assembly sequence with} \\ S_{\alpha_k} = S, \text{ and } \alpha_k \text{ is } \mathcal{T}\text{-terminal} \end{array} \right\}.$$

$$\exists\forall\text{CNF-UNSAT} = \left\{ \langle \varphi \rangle \mid \begin{array}{l} \varphi \text{ is a true quantified Boolean} \\ \text{formula } \varphi = \exists x\forall y\neg\phi(x, y), \\ \text{where } \phi \text{ is an unquantified} \\ \text{CNF formula with } n + m \\ \text{input bits } x = x_1, \dots, x_n \\ \text{and } y = y_1, \dots, y_m \end{array} \right\}.$$

Clearly $\text{MINTILESET}_V \in P$. $\text{MINTILESET} \in \Sigma_2^P$ because $\langle S, c \rangle \in \text{MINTILESET}$ if and only if there exists $\mathcal{T} = (T, \sigma, 2)$ with $|T| \leq c$ such that for all \mathcal{T} -assembly sequences $\vec{\alpha} = (\sigma, \alpha_2, \dots, \alpha_k)$ of length $k = |S|$, $\langle S, c, \mathcal{T}, \vec{\alpha} \rangle \in \text{MINTILESET}_V$, with $|\langle \mathcal{T} \rangle|$ and $|\langle \vec{\alpha} \rangle|$ bounded by $O(|\langle S, c \rangle|^2)$.

To show that MINTILESET is Σ_2^P -hard, we show that $\exists\forall\text{CNF-UNSAT} \leq_m^P \text{MINTILESET}$, where

We follow a similar strategy to the reduction of 3SAT to MINDIRTILESET shown in [5]. The \leq_m^P -reduction $\langle \varphi \rangle \mapsto \langle S, c \rangle$ works as follows. First, we compute a tree $\Upsilon \in \mathcal{FS}$ that “represents” φ with subtree gadgets that encode possible variable assignments and their effect on clauses. We then process Υ with the polynomial-time algorithm described in [5] that computes the minimum number of tile types needed to strictly self-assemble a tree. Let $\mathcal{T} = (T, \sigma, 2)$ be this minimal TAS that strictly

self-assembles Υ , and let $c = |T|$. We then compute a shape $S \in \mathcal{FS}$ such that $\Upsilon \subset S$ with the property that, if φ is true, then the tile types in T can be modified, solely through changing some null glues to be single or double glues, producing a TAS $\mathcal{T}' = (T', \sigma, 2)$ with $|T'| = |T| = c$ such that \mathcal{T}' strictly self-assembles S , and if φ is false, then no TAS with at most c tile types can strictly self-assemble S . The shape S is shown in Figure 3. In Figure 3, the height of pillars is set to a number bigger than 20ℓ , where ℓ is the number of variables in φ .⁹

Suppose that ϕ has k clauses C_1, \dots, C_k and $\ell = n + m$ input variables v_1, \dots, v_ℓ , where $v_1, \dots, v_n = x_1, \dots, x_n$ are the \exists -variables of φ and $v_{n+1}, \dots, v_\ell = y_1, \dots, y_m$ are the \forall -variables of φ . A clause C is *satisfied* by variable v if C contains literal v and v is true, or if C contains literal $\neg v$ and v is false. For each $1 \leq i \leq k$ and $1 \leq j \leq \ell$, define the following six gadgets:

1. SST_{ij} : C_i is satisfied by v_p for some $1 \leq p < j$, and v_j is true.
2. SSF_{ij} : C_i is satisfied by v_p for some $1 \leq p < j$, and v_j is false.
3. UUT_{ij} : C_i is unsatisfied by v_p for every $1 \leq p \leq j$, and v_j is true.
4. UUF_{ij} : C_i is unsatisfied by v_p for every $1 \leq p \leq j$, and v_j is false.
5. UST_{ij} : C_i is unsatisfied by v_p for every $1 \leq p < j$, C_i is satisfied by v_j , and v_j is true.
6. USF_{ij} : C_i is unsatisfied by v_p for every $1 \leq p < j$, C_i is satisfied by v_j , and v_j is false.

Each of these six main varieties of “information-bearing” gadgets is shown in Figure 4. Each gadget is designed to minimize the amount of “potential unwanted cooperative strength-1 binding” when they are placed next to each other in the “matrix” of gadgets in the upper left of Figure 3.¹⁰ Each gadget encodes the integers i and j , as well as encoding the information about the clause C_i and variable v_j as described above. Some of the “boundary case” gadgets are shaped slightly

differently than those in Figure 4. If $i = k$ (a “top gadget”), the top of the gadget will not encode information about the truth value of the variable v_i . If $j = \ell$ (a “right gadget”), the right side of the gadget will still encode whether the clause is satisfied, but the gadget will have a different shape than for $1 \leq j < \ell$. These special boundary shapes are shown in Figure 3.

Not all six varieties of gadgets are created for each (i, j) ; the only gadgets created are those that are logically consistent with some variable assignment to ϕ . The matrix and pillars portion of S (i.e., $S \setminus \Upsilon$) depends only on the number of \exists -variables, the number of \forall -variables, and the number of clauses. The remainder of the information about φ is encoded in the following choices about which gadgets to create in Υ . In the case of $j = 1$, the gadgets SST_{i1} and SSF_{i1} are not created. For any clause C_i in which the literal v_j (resp. $\neg v_j$) does not appear, the gadget UST_{ij} (resp. USF_{ij}) is not created. Similarly, for any clause C_i in which *no* literal v_p (resp. $\neg v_p$) appears for any $1 \leq p < j$, the gadget SST_{ij} (resp. SSF_{ij}) is not created. Finally, for any clause C_i in which the literal v_j (resp. $\neg v_j$) *does* appear, the gadget UUT_{ij} (resp. UUF_{ij}) is not created.

The tree Υ is S without the “matrix” on the top left and the “pillars” beneath it that connect it to the bottom row. Let $c = C^{\text{tc}}(\Upsilon)$. We assume that the seed is placed on the rightmost position of the bottom row, for both the shapes Υ and S . In the full version of this paper we show how to modify the shapes to enforce this restriction. Briefly, one can create two copies of S joined at their rightmost points by a short “bridge” to enforce that the minimal tile set for this new shape reuses two copies of the minimal tile set for S subject to the constraint of placing the seed on the right, with the seed necessarily placed somewhere in the bridge.

The steps needed to complete the proof are divided into several lemmas, stated here without proof. The proofs of Lemmas 4.1 - 4.4 are provided in the full version of this extended abstract. After stating the lemmas, we conclude the proof sketch with a brief intuitive overview of the technique used in the full version.

LEMMA 4.1. *If φ is true, then $C^{\text{tc}}(S) \leq c$.*

Assume that $\mathcal{T}_S = (T_S, \sigma, 2)$ is a TAS that strictly self-assembles S with the seed placed at the rightmost position on the bottom row.

LEMMA 4.2. *If $|T_S| \leq c = C^{\text{tc}}(\Upsilon)$, then there is a TAS $\mathcal{T}_\Upsilon = (T_\Upsilon, \sigma, 2)$ that can be obtained from $\mathcal{T}_S = (T_S, \sigma, 2)$ by only changing a number of double glues to null glues (in particular implying that $|T_\Upsilon| = |T_S|$), such that \mathcal{T}_Υ strictly self-assembles Υ .*

⁹Actually, it is enough to set the height of pillars to any number bigger than the width of the clause-variable matrix.

¹⁰Strength-1 glues can only have an effect on growth of gadgets in the matrix when they are on tiles on the gray positions r, s , and t in Figure 4, if the tile types used to assemble those gadgets in the matrix are the same as those used to assemble the gadgets in Υ . This is useful in proving the converse direction of the reduction by showing that if a tile assembly system with $\leq c$ tile types strictly self-assembles S , then φ must be true.

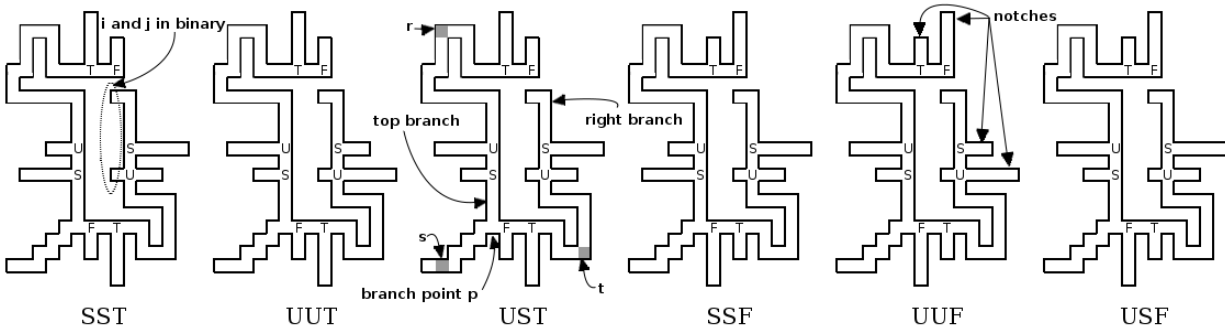


Figure 4: Six main varieties of “information-bearing” tree gadgets used in the reduction. The position (i, j) where the gadget is intended to go in the matrix is encoded in binary. Gadgets intended for the top row are missing the top “T/F” bumps, and gadgets intended for the right column are different on the right depending on whether the clause is satisfied or not, as shown in Figure 3.

$B \subset \mathbb{Z}^2$ represents the subshape of S that does not have the black matrix on the left, but has the pillars beneath it, and $I \subset \mathbb{Z}^2$ denotes the set of $k \times \ell$ positions (where k is the number of clauses in ϕ and ℓ is the number of variables) marked by small circles in Figure 3. A set $I' \subseteq I$ is called a *staircase* if the following implication holds:

$$[(x_1, y_1) \in I', (x_2, y_2) \in I, x_2 \leq x_1, \text{ and } y_2 \leq y_1] \implies (x_2, y_2) \in I'.$$

The following lemma states the “inductive step” of the proof of Lemma 4.4. Namely, if gadgets of a minimal tile system for S grow to fill in part of the matrix “in the way we intend”, then the only way to extend this growth to fill in an additional gadget is also “in the way we intend.” In the following lemma, “right branch” and “top branch” refer to the two subtrees of a gadget rooted at the branch as shown in Figure 4.

LEMMA 4.3. *Suppose \mathcal{T}_S has at most $c = C^{tc}(\Upsilon)$ tile types. Let $\alpha \in \mathcal{A}[\mathcal{T}_S]$ be a producible assembly such that*

1. $B \subseteq S_\alpha \subseteq S$. (where B is the subshape of S that does not have the black matrix on the left, but has the pillars beneath it)
2. $S_\alpha \cap I$ is a staircase of cardinality $m < |I|$.
3. All tile types in $\alpha(S_\alpha \cap I)$ are branch tiles of gadgets.
4. The right and top branches of tiles in $S_\alpha \cap I$ are present in α .
5. If there exists $p \in I$ such that $p \notin S_\alpha$, then $S_\alpha \cap (p + G^*) = \emptyset$.

Then there exists an assembly $\beta \in \mathcal{A}[\mathcal{T}_S]$ such that $\alpha \rightarrow \beta$ and requirements (1)-(5) are satisfied with “ α ” replaced by “ β ” and “ m ” replaced by “ $m + 1$ ”.

The following lemma follows (with a bit of effort) from the previous two.

LEMMA 4.4. *If \mathcal{T}_S has at most c tile types, then φ is true.*

Lemmas 4.1 and 4.4 establish each direction of the claim that φ is true $\iff C^{tc}(S) \leq c$. Intuitively, since Υ is a “tree-like” subshape of S (despite the leftmost tiles intersecting cycles in S), any tile system that strictly self-assembles S must place tiles in the bottom row that do not appear anywhere else in Υ . φ is true $\implies C^{tc}(S) \leq c$ because we can modify the null glues of tiles in the left half of the bottom row of Υ to be double glues matching those tile types from the pillars on the right to grow the pillars on the left. In the case of the \exists -variables x we choose an assignment by our choice of double glues. In the case of \forall -variables y we have no choice; we must allow both the “false” and “true” pillars to grow and nondeterministically compete to assign a bit to each y_i . We can then modify null glues in the gadgets and pillars to be single glues that propagate information about the neighbors of a gadget to be placed in the matrix. Therefore the assembly of the matrix “evaluates $\phi(x, y)$ ” and if it is false, strictly self-assembles S . The reverse direction is more tedious to establish. Again, since Υ is a “tree-like” subshape of S , any TAS strictly self-assembling S already uses c tile types just to assemble the Υ portion of S (derived from Lemma 4.2). Therefore to assemble all of S using c tile types requires reusing these same tile types. Our gadget design, together with the properties of minimal tile sets for trees, allow us to conclude that the only way to tile the matrix is “using the gadgets in the way they were intended”, which means the rightmost vertical bar

of the matrix cannot form unless at least one clause is not satisfied; i.e., φ is true. \square

5 Conclusion

We have investigated the power of nondeterminism for the strict self-assembly of shapes in the abstract Tile Assembly Model. We showed that for both the infinite and finite cases, even when the shape is required to be strictly self-assembled, nondeterminism can help to assemble the shape, by making strict self-assembly possible in the infinite case, and reducing tile complexity in the finite case. Furthermore, the problem of finding the minimum tile set to strictly self-assemble a shape is strictly harder (in the sense of nondeterministic time complexity) than that of finding the minimum directed tile set that does so, unless $\text{NP} = \Sigma_2^P$.

There are some interesting questions that remain open:

1. What is the fastest growing function $f : \mathbb{N} \rightarrow \mathbb{N}$ for which one could prove a statement of the form “For infinitely many $n \in \mathbb{N}$, there is a finite shape $S \subset \mathbb{Z}^2$ such that $C^{\text{tc}}(S) \leq n$ and $C^{\text{dtc}}(S) \geq f(n)$ ”? The proof of Theorem 4.1 of the present paper establishes this statement for $f(n) = 1.4999n$. Can $f(n)$ be made, for example, n^2 or 2^n ? What is an upper bound for f above which such a statement is false? Note that Theorem 3.1 establishes such a statement for *all* functions $f : \mathbb{N} \rightarrow \mathbb{N}$ if the shape is allowed to be infinite. However, when designing complex tile systems, a common challenge is to direct a group of tiles to stop growing,¹¹ so it would be interesting to identify a family of *finite* shapes with a fast-growing gap between the two tile complexity measures. This would imply that sometimes it *really* helps to employ nondeterminism.
2. We have showed that the optimization problem of finding precisely the smallest number of tile types to strictly self-assemble a shape is Σ_2^P -hard. Can it be shown that for some $\alpha > 1$, the solution is Σ_2^P -hard to approximate within multiplicative factor α ? This may be related to Question 1.
3. Is there an $\alpha > 1$ such that it is NP-hard to find an α -approximate solution to the minimum directed tile set problem?

¹¹For example, $C^{\text{tc}}(S) \approx C^{\text{dtc}}(S) = O(\log n / \log \log n)$ for S an $n \times k$ rectangle with $n \geq k \geq \log n / \log \log n$, but $C^{\text{tc}}(S)$ and $C^{\text{dtc}}(S)$ increase steadily towards n as k decreases from $\log n / \log \log n$ to 1; “counting” to the length of the rectangle and then stopping becomes more difficult as the rectangle’s width decreases.

4. Shape-building is one common goal of self-assembly; pattern-painting is another. In particular, it is possible to assemble some patterns, such as disconnected sets, if we change the definition of what is interpreted as the assembled object. We say that a TAS $\mathcal{T} = (T, \sigma, 2)$ *weakly self-assembles* a set $S \subseteq \mathbb{Z}^2$ if there is a subset $B \subseteq T$ (the tile types that are “painted black”) such that, for all $\alpha \in \mathcal{A}_{\square}[\mathcal{T}]$, $\alpha^{-1}(B) = S$. In other words, the set of positions with a black tile is guaranteed to be S . In the case $B = T$, this definition is equivalent to strict self-assembly, but for $B \subsetneq T$ the shape is allowed to grow outside the desired pattern using tile types from $T \setminus B$ to allow “extra computation room” for painting the pattern using tile types from B . Such a definition is appropriate for modeling practical goals such as self-assembled circuit layouts [21, 25, 28, 31, 32, 46] or placement of guides for walking molecular robots [26]; see [22, 23] for more discussion of the theoretical issues of weak self-assembly. It remains open to prove or disprove analogs of Theorems 3.1 and 4.1, with “weakly” substituted for “strictly”. In other words, is it possible to uniquely paint an infinite (resp. finite) pattern with a tile system, but every tile system that does so (resp. that does so with no extra tile types) is not directed?

5. It remains open to prove or disprove analogs of Theorems 3.1 and 4.1, with “weakly” substituted for “strictly” and with “strict” substituted for “directed”. In other words, is it possible to uniquely paint an infinite (resp. finite) pattern with a tile system, but every tile system that does so (resp. that does so with no extra tile types) must self-assemble more than one shape on which this pattern is painted?

6. What is the status of the optimization problem of determining the minimum number of tile types to weakly self-assemble a finite pattern? Let $\mathcal{F}(\mathbb{Z}^2) \subset \mathcal{P}(\mathbb{Z}^2)$ denote the set of all finite subsets of \mathbb{Z}^2 . Define

$$\text{MINWEAKTILESET} = \left\{ \langle S, c \rangle \mid \begin{array}{l} S \in \mathcal{F}(\mathbb{Z}^2), c \in \mathbb{Z}^+, \text{ and} \\ (\exists \mathcal{T} = (T, \sigma, 2)) |T| \leq c \text{ and} \\ \mathcal{T} \text{ weakly self-assembles } S \end{array} \right\}.$$

It is not obvious whether MINWEAKTILESET is even contained in PSPACE or EXP, for instance.

7. What is the status of MINDIRECTEDWEAKTILESET, defined similarly to MINWEAKTILESET but also requiring \mathcal{T} to be directed?

8. In [5] the authors show that for the special cases of tree and square shapes, the minimum directed tile set problem is in P. For trees, it is straightforward to verify that the minimum tile set is always directed, so the answer is the same whether or not we restrict attention to directed tile sets. What is the complexity of the minimum tile set problem restricted to squares? The polynomial-time algorithm given in [5] crucially depends on the existence of a polynomial-time algorithm for the *directed shape verification* problem of determining whether a given tile system strictly self-assembles a given shape and is directed. Removing the directed constraint on this shape verification problem, even when restricted to the case of squares, makes the problem coNP-complete [6, 18, 24]. Perhaps this means that the minimum tile set problem restricted to squares is hard as well. On the other hand, since this problem is sparse,¹² Fortune’s Theorem [16] implies that it cannot be coNP-hard (nor NP-hard by Mahaney’s Theorem [27]) unless P = NP.

References

- [1] Scott Aaronson and John Watrous. Closed timelike curves make quantum and classical computing equivalent. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science*, 465(2102):631, 2009.
- [2] Zachary Abel, Nadia Benbernou, Mirela Damian, Erik D. Demaine, Martin Demaine, Robin Flatland, Scott Kominers, and Robert Schweller. Shape replication through self-assembly and RNase enzymes. In *SODA 2010: Proceedings of the Twenty-first Annual ACM-SIAM Symposium on Discrete Algorithms*, Austin, Texas, 2010. Society for Industrial and Applied Mathematics.
- [3] Leonard M. Adleman. Towards a mathematical theory of self-assembly. Technical report, University of Southern California, 2000.
- [4] Leonard M. Adleman, Qi Cheng, Ashish Goel, and Ming-Deh Huang. Running time and program size for self-assembled squares. In *STOC 2001: Proceedings of the thirty-third annual ACM Symposium on Theory of Computing*, pages 740–748, Hersonissos, Greece, 2001. ACM.
- [5] Leonard M. Adleman, Qi Cheng, Ashish Goel, Ming-Deh A. Huang, David Kempe, Pablo Moisset de Espanés, and Paul W. K. Rothmund. Combinatorial optimization problems in self-assembly. In *STOC 2002: Proceedings of the Thirty-Fourth Annual ACM Symposium on Theory of Computing*, pages 23–32, 2002.
- [6] Gagan Aggarwal, Qi Cheng, Michael H. Goldwasser, Ming-Yang Kao, Pablo Moisset de Espanés, and Robert T. Schweller. Complexities for generalized models of self-assembly. *SIAM Journal on Computing*, 34:1493–1515, 2005. Preliminary version appeared in SODA 2004.
- [7] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.
- [8] Robert D. Barish, Rebecca Schulman, Paul W. Rothmund, and Erik Winfree. An information-bearing seed for nucleating algorithmic self-assembly. *Proceedings of the National Academy of Sciences*, 106(15):6054–6059, March 2009.
- [9] 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.
- [10] Harish Chandran, Nikhil Gopalkrishnan, and John H. Reif. The tile complexity of linear assemblies. In *ICALP 2009: 36th International Colloquium on Automata, Languages and Programming*, volume 5555, pages 235–253. Springer, 2009.
- [11] Erik D. Demaine, Martin L. Demaine, Sándor P. Fekete, Mashhood Ishaque, Eynat Rafalin, Robert T. Schweller, and Diane L. Souvaine. Staged self-assembly: Nanomanufacture of arbitrary shapes with $O(1)$ glues. *Natural Computing*, 7(3):347–370, 2008.
- [12] Erik D. Demaine, Matthew J. Patitz, Robert T. Schweller, and Scott M. Summers. Self-assembly of arbitrary shapes with RNA and DNA tiles. Technical Report 1004.4383, Computing Research Repository, 2010.
- [13] David Doty. Randomized self-assembly for exact shapes. In *FOCS 2009: Proceedings of the Fiftieth IEEE Conference on Foundations of Computer Science*, 2009.
- [14] David Doty, Lila Kari, and Benoît Masson. Negative interactions in irreversible self-assembly. In *16th International Meeting on DNA Computing (DNA 16)*, Hong Kong, China, June 14–17, 2010., 2010.
- [15] David Doty, Matthew J. Patitz, and Scott M. Summers. Limitations of self-assembly at temperature 1. *Theoretical Computer Science*. to appear. Preliminary version appeared in DNA 15.
- [16] Steven Fortune. A note on sparse complete sets. *SIAM Journal on Computing*, 8(3):431–433, 1979.
- [17] Kenichi Fujibayashi, Rizal Hariadi, Sung Ha Park, Erik Winfree, and Satoshi Murata. Toward reliable algorithmic self-assembly of DNA riles: A fixed-width

¹²More precisely, if one takes a bit of care in encoding the problem, then it can be assumed sparse. Assume that each square S has its lower left-corner at the origin and is encoded by a list of its points in binary using precisely $\lfloor \log |S| \rfloor + 1$ bits for each point, and the bound c on the number of tile types is also encoded in binary using precisely $\lfloor \log |S| \rfloor + 1$ bits. Then a sparse set that is \equiv_m^P -equivalent to the minimum tile set problem for squares is $\{\langle S, c \rangle \in \text{MIN-TILESET} \mid S \text{ is an } n \times n \text{ square with lower-left corner } (0, 0), \text{ and } 1 \leq c \leq n^2\}$. In fact we can even require $c \leq O(\log n / \log \log n)$ by a result of [4], but the trivial upper bound of n^2 suffices to obtain sparseness.

- cellular automaton pattern. *Nano Letters*, 8(7):1791–1797, 2007.
- [18] Michael R. Garey and David S. Johnson. *Computers and Intractability*. W. H. Freeman, New York, 1979.
- [19] Ming-Yang Kao and Robert T. Schweller. Reducing tile complexity for self-assembly through temperature programming. In *SODA 2006: Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 571–580, 2006.
- [20] Ming-Yang Kao and Robert T. Schweller. Randomized self-assembly for approximate shapes. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Haldrsson, Anna Ingólfssdóttir, and Igor Walukiewicz, editors, *ICALP 2008: International Colloquium on Automata, Languages, and Programming*, volume 5125 of *Lecture Notes in Computer Science*, pages 370–384. Springer, 2008.
- [21] Ryan J. Kershner, Luisa D. Bozano, Christine M. Micheel, Albert M. Hung, Ann R. Fornof, Jennifer N. Cha, Charles T. Rettner, Marco Bersani, Jane Frommer, Paul W. K. Rothmund, and Gregory M. Wallraff. Placement and orientation of individual DNA shapes on lithographically patterned surfaces. *Nature Nanotechnology*, 3:557–561, 2009.
- [22] James I. Lathrop, Jack H. Lutz, Matthew J. Patitz, and Scott M. Summers. Computability and complexity in self-assembly. *Theory of Computing Systems*. to appear.
- [23] James I. Lathrop, Jack H. Lutz, and Scott M. Summers. Strict self-assembly of discrete Sierpinski triangles. *Theoretical Computer Science*, 410:384–405, 2009.
- [24] Harry R. Lewis and Christos H. Papadimitriou. *Elements of the Theory of Computation*. Prentice-Hall, Inc., Upper Saddle River, New Jersey, 1998.
- [25] Dage Liu, Sung Ha Park, John H. Reif, and Thomas H. LaBean. DNA nanotubes self-assembled from triple-crossover tiles as templates for conductive nanowires. *PNAS: Proceedings of the National Academy of Sciences of the United States of America*, 101(3):717, 2004.
- [26] Kyle Lund, Anthony T. Manzo, Nadine Dabby, Nicole Micholotti, Alexander Johnson-Buck, Jeanette Nangreave, Steven Taylor, Renjun Pei, Milan N. Stojanovic, Nils G. Walter, Erik Winfree, and Hao Yan. Molecular robots guided by prescriptive landscapes. *Nature*, 465:206–210, 2010.
- [27] Stephen R. Mahaney. Sparse complete sets for NP: Solution of a conjecture of Berman and Hartmanis. *JCSS: Journal of Computer and System Sciences*, 25, 1982. Preliminary version appeared in FOCS 1980.
- [28] Hareem T. Maune, Si-Ping Han, Robert D. Barish, Marc Bockrath, William A. Goddard III, Paul W. K. Rothmund, and Erik Winfree. Self-assembly of carbon nanotubes into two-dimensional geometries using DNA origami templates. *Nature Nanotechnology*, 5:61–66, 2010.
- [29] Ján Mañuch, Ladislav Stacho, and Christine Stoll. Step-assembly with a constant number of tile types. In *ISAAC 2009: Proceedings of the 20th International Symposium on Algorithms and Computation*, pages 954–963, Berlin, Heidelberg, 2009. Springer-Verlag.
- [30] Ján Mañuch, Ladislav Stacho, and Christine Stoll. Two lower bounds for self-assemblies at temperature 1. *Journal of Computational Biology*, 17(6):841–852, 2010.
- [31] Sung Ha Park, Constantin Pistol, Sang Jung Ahn, John H. Reif, Alvin R. Lebeck, Chris Dwyer, and Thomas H. LaBean. Finite-Size, Fully Addressable DNA Tile Lattices Formed by Hierarchical Assembly Procedures. *Angewandte Chemie-International Edition*, 45(40):6607–6607, 2006.
- [32] Sung Ha Park, Hao Yan, John H. Reif, Thomas H. LaBean, and Gleb Finkelstein. Electronic Nanostructures Templated on Self-Assembled DNA Scaffolds. *Nanotechnology*, 15:S525–S527, 2004.
- [33] Matthew J. Patitz. Simulation of self-assembly in the abstract tile assembly model with ISU TAS. In *FNANO 2009: 6th Annual Conference on Foundations of Nanoscience: Self-Assembled Architectures and Devices (Snowbird, Utah, USA, April 20-24 2009)*, pages 209–219. Sciencetechnica, 2009.
- [34] Paul W. K. Rothmund. *Theory and Experiments in Algorithmic Self-Assembly*. PhD thesis, University of Southern California, December 2001.
- [35] Paul W. K. Rothmund and Erik Winfree. The program-size complexity of self-assembled squares (extended abstract). In *STOC 2000: Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing*, pages 459–468, 2000.
- [36] Paul W.K. Rothmund, Nick Papadakis, and Erik Winfree. Algorithmic self-assembly of DNA Sierpinski triangles. *PLoS Biology*, 2(12):2041–2053, 2004.
- [37] Marcus Schaefer and Christopher Umans. Completeness in the polynomial-time hierarchy: Part I: A compendium. *SIGACTN: SIGACT News (ACM Special Interest Group on Automata and Computability Theory)*, 33(3):32–49, September 2002.
- [38] Nadrian C. Seeman. Nucleic-acid junctions and lattices. *Journal of Theoretical Biology*, 99:237–247, 1982.
- [39] David Soloveichik and Erik Winfree. Complexity of self-assembled shapes. *SIAM Journal on Computing*, 36(6):1544–1569, 2007. Preliminary version appeared in DNA 10.
- [40] Larry J. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3(1):1–22, October 1976.
- [41] Scott M. Summers. Reducing tile complexity for the self-assembly of scaled shapes through temperature programming. Technical Report 0907.1307, Computing Research Repository, 2009.
- [42] Hao Wang. Proving theorems by pattern recognition – II. *The Bell System Technical Journal*, XL(1):1–41, 1961.
- [43] Hao Wang. Dominoes and the AEA case of the decision problem. In *Proceedings of the Symposium on Mathematical Theory of Automata (New York, 1962)*, pages 23–55. Polytechnic Press of Polytechnic Inst. of

Brooklyn, Brooklyn, N.Y., 1963.

- [44] Erik Winfree. *Algorithmic Self-Assembly of DNA*. PhD thesis, California Institute of Technology, June 1998.
- [45] Celia Wrathall. Complete sets and the polynomial-time hierarchy. *Theoretical Computer Science*, 3(1):23–33, October 1976.
- [46] Hao Yan, Sung Ha Park, Gleb Finkelstein, John H. Reif, and Thomas H. LaBean. DNA-templated self-assembly of protein arrays and highly conductive nanowires. *Science*, 301(5641):1882, 2003.