THEORY OF COMPUTING, Volume 9 (1), 2013, pp. 1–29
www.theoryofcomputing.org

The Power of Nondeterminism in Self-Assembly*

Nathaniel Bryans† Ehsan Chiniforooshan‡ David Doty§ Lila Kari¶
Shinnosuke Seki‖

Received February 25, 2011; Revised October 3, 2012; Published January 21, 2013

Abstract: We investigate the role of nondeterminism in Winfree's abstract Tile Assembly Model (aTAM), in particular how its use in tile systems affects the resource requirements. We show that for infinitely many $c \in \mathbb{N}$, there is a finite shape S that is self-assembled by a tile system (meaning that all of the various terminal assemblies produced by the tile system have shape S) with c tile types, but every *directed* tile system that self-assembles S (i. e., has only one terminal assembly, whose shape is S) needs more than c tile types. We extend the technique to prove our main theorem, that the problem of finding the minimum number of

*An extended abstract of this paper appeared as [9] in SODA'11.

†This author was supported by the NSERC Undergraduate Student Research Awards (USRA) grant.

‡This author was supported by NSERC Discovery Grant R2824A01 and the Canada Research Chair Award in Biocomputing to Lila Kari.

§This author was supported by NSERC Discovery Grant R2824A01 and the Canada Research Chair Award in Biocomputing to Lila Kari, by a Computing Innovation Fellowship under NSF grant 1019343 and NSF grants CCF-1219274 and CCF-1162589 and the Molecular Programming Project under NSF grant 0832824.

¶This author was supported by NSERC Discovery Grant R2824A01 and the Canada Research Chair Award in Biocomputing.

‖This author was supported by NSERC Discovery Grant R2824A01 and the Canada Research Chair Award in Biocomputing to Lila Kari, by Kyoto University Start-up Grant-in-Aid for Young Scientists (No. 021530), and by HIIT Pump Priming Project 902184/T30606.

ACM Classification: F.2.2

AMS Classification: 68Q17

Key words and phrases: nondeterminism, polynomial hierarchy, completeness, self-assembly

© 2013 Nathaniel Bryans, Ehsan Chiniforooshan, David Doty, Lila Kari, and Shinnosuke Seki
Licensed under a Creative Commons Attribution License (CC-BY) DOI: 10.4086/toc.2013.v009a001

tile types that self-assemble a given finite shape is Σ_2^{P}-complete. We then show an analogous "computability theoretic" result: there is an infinite shape S that is self-assembled by a tile system but not by any directed tile system.

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 solution, they are guaranteed to bind so as to 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 [42] 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 [38], 1.4% in 2007 [20], 0.13% in 2009 [7], and 0.05% in 2010 [17]. Erik Winfree [49] introduced the abstract Tile Assembly Model (aTAM)—based on a constructive version of Wang tiling [47, 48]—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 [39] investigated a self-assembly resource bound known as *tile complexity*, the minimum number of tile types needed to self-assemble a shape. They showed that for most n, the problem of self-assembling an $n \times n$ square has tile complexity $\Omega(\log(n)/\log\log(n))$, and Adleman, Cheng, Goel, and Huang [3] exhibited a construction showing that this lower bound is asymptotically tight. We discuss two different models of self-assembly below (unique versus strict), but both of the previous results hold for either model. Under natural generalizations of the model [1, 5, 8, 10–14, 19, 22, 23, 32, 44, 46], tile complexity can be reduced for tasks such as square-building and self-assembly of more general shapes.

We now briefly describe the aTAM; a formal definition is given in Section 2. A *tile* in the aTAM is a unit square with a kind and strength of "glue" on each of its sides. Tiles are assumed not to rotate so that each side has a well-defined direction: north, south, east, or west. A *tile assembly system* $\mathcal{T} = (T, \sigma, \tau)$ consists of a finite set T of tile types (with infinitely many tiles of each type in T available), a *seed tile* $\sigma \in T$, from which growth is assumed to nucleate, and a *temperature* τ, assumed to be 2 in this paper. Self-assembly proceeds from the seed tile σ, with tiles of types in T successively attaching themselves to the existing assembly. Two tiles placed next to each other *interact* if the glues on their abutting sides match, and a tile *binds* to a binding site on an assembly if the total strength on all of its interacting sides is at least τ. The choice of which binding site to attach is nondeterministic. It is possible that a single binding site could have more than one tile type able to attach with strength τ, which is also a choice made nondeterministically (the impossibility of this situation occurring precisely characterizes the "directed" tile systems, described informally next and defined formally in Section 2).

Since we use the term "nondeterministic" informally in various contexts to indicate any situation in which more than one choice is available, to avoid ambiguity, we introduce new terms to distinguish precisely between the two specific types of determinism we consider in this paper. We say a tile system is *directed* if it is guaranteed to form one unique terminal *assembly* (terminal meaning that no tiles can attach to it), 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. In this case, we say the tile system *uniquely self-assembles* the shape (since the terminal assembly is unique). We say a tile system *strictly self-assembles* a *shape* (and we say the tile system is *strict*) if all of its terminal assemblies are guaranteed to have that shape. In the case that there is more than one terminal assembly with the same shape, these various assemblies will have different tile types at the same position (but will be in agreement about which positions have a tile). A strict tile system may not be directed, despite enforcing another sort of determinism by guaranteeing what shape is to form. Figure 1 shows an example of a non-directed temperature tile system that strictly self-assembles a 2×2 square.

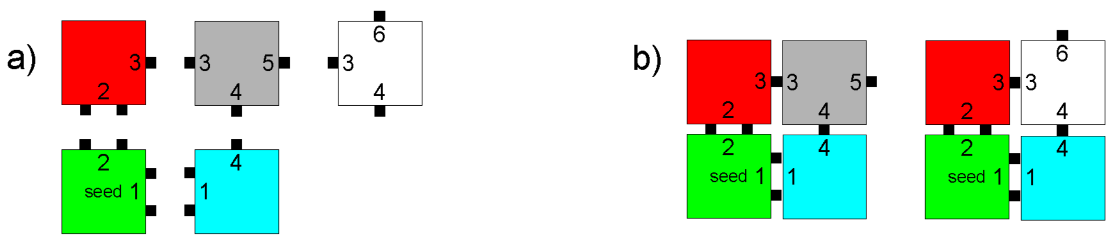

Figure 1: a) Example of a non-directed temperature-2 tile system that strictly self-assembles a 2×2 square. The seed tile is labeled as such, and the strength of a glue on a tile type's side is indicated by the number of small black squares. b) The two terminal assemblies (both having the same shape) of the tile system. In this example, removing a single tile type (for instance, the white tile type), makes the tile system directed without affecting its ability to strictly self-assemble a 2×2 square.

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 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 self-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 [44] as well as lower bounds and impossibility proofs [5, 16, 31] 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.

1.1 Main results of this paper

We show that for infinitely many $c \in \mathbb{N}$ there is a finite shape S that is strictly self-assembled by a tile system with c tile types, but (unlike the example in Figure 1) every directed tile system that strictly self-assembles S has more than c tile types. In fact, to strictly self-assemble the shape in a directed tile system requires more than $\approx \frac{3}{2}c$ tile types. Schweller [41] has improved this by exhibiting a family of shapes with super-linear gap between their directed and non-directed tile complexities. The issue is discussed in more detail in Section 5. The techniques are used in the proof of our main theorem, which shows that (the decision version of) the problem of finding the minimum number of tile types that strictly self-assemble a given finite shape is complete for the complexity class $\Sigma_2^{\mathsf{P}} = \mathsf{NP}^{\mathsf{NP}}$. In contrast, the problem of finding the minimum size *directed* tile system that strictly self-assembles a shape was shown to be NP-complete by Adleman, Cheng, Goel, Huang, Kempe, Moisset de Espanés, and Rothemund [4]. (This result is discussed and compared to our own in more detail below.) That is, not only is the answer to the question "What is the tile complexity of shape S?" different from the answer to "What is the directed tile complexity of shape S?", but the former question is fundamentally more difficult to answer than the latter, barring an unlikely collapse of complexity classes.

We then show an analogous phenomenon in the infinitary case: 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, nondirectedness allows certain shapes to be algorithmically self-assembled that are totally "unassemblable" (to borrow Adleman's term analogous to "uncomputable" [2]) under the constraint of directedness.

Based on these results, we conclude that nondeterminism in tile type placement confers extra power to self-assemble a shape from a small tile system, but unless the polynomial hierarchy collapses, it is computationally more difficult[1] to exploit this power by finding the size of the smallest tile system, compared to finding the size of the smallest directed tile system.

1.2 Comparison with related work

As indicated above, Adleman, Cheng, Goel, Huang, Kempe, Moisset de Espanés, and Rothemund [4] consider the optimization problem, given a finite shape S, what is the size of the smallest tile system that uniquely self-assembles S (i. e., smallest directed tile system that strictly self-assembles S)? We consider the optimization problem, given a finite shape S, what is the size of the smallest tile system that strictly self-assembles S? In other words, our optimization problem considers strictly more tile systems, by allowing those that self-assemble more than one terminal assembly, so long as all of those assemblies

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

have the same shape, whereas in [4] the optimization problem considers only directed tile systems, those that self-assemble a unique terminal assembly. Therefore, for any given shape S, the answer to our optimization problem will be less than or equal to the answer to the optimization problem of [4].

A related result was shown by Aggarwal, Cheng, Goldwasser, Kao, Moisset de Espanes, and Schweller [5], who studied a different problem using the same model of shape assembly that we use (strict self-assembly). Theirs is a verification problem: given a finite shape S *and* a tile system $\mathcal{T}$, does $\mathcal{T}$ strictly self-assemble S? They showed this problem to be coNP-complete. This is in contrast to the equivalent verification problem for unique assembly (i. e., strict self-assembly by a directed tile system, the model considered in [4]), which was shown in [4] to be solvable in polynomial time.

	Strict self-assembly	Unique self-assembly
Verification	coNP-complete [5]	P [4]
Optimization	Σ_2^P-complete [this paper]	NP-complete [4]

Table 1: Four combinatorial problems in self-assembly of shapes, based on two binary choices: 1) *verification* that a given tile system self-assembles a given shape versus *optimization* of the number of tile types required to self-assemble a given shape, and 2) *strict* self-assembly (many terminal assemblies allowed but they must all have the same shape) versus *unique* self-assembly (only one terminal assembly allowed).

These four results are outlined in Table 1. It is instructive to observe why the containments in NP and Σ_2^P of the bottom row of Table 1 hold. In the directed case, one can nondeterministically guess a tile system $\mathcal{T}$ and then use the polynomial-time algorithm of the upper-right entry of Table 1 to check whether $\mathcal{T}$ is uniquely self-assembles S. In the non-directed case, we can also nondeterministically guess a tile system $\mathcal{T} = (T, \sigma, \tau)$. The polynomial-time algorithm for unique self-assembly verification does not work for strict self-assembly; indeed, that problem is coNP-complete as indicated in the upper-left entry of the table. Therefore the optimization problem for strict self-assembly is not obviously contained in NP. To prove containment in Σ_2^P, it suffices to use a universal polynomially-bounded quantifier to check all possible sequences of tile additions to the seed of $\mathcal{T}$ to verify that all terminal assemblies that result have the shape S. Since $|T| \leq |S|$ if $\mathcal{T}$ is a minimal tile system for S, the length in bits of the candidate tile addition sequences is bounded by a polynomial in $|S|$.

2 Abstract Tile Assembly Model

This section gives a terse definition of the abstract Tile Assembly Model (aTAM, [49]). This is not a tutorial; for readers unfamiliar with the aTAM, Rothemund and Winfree [39] give an excellent introduction to the model. Doty [15] provides a high-level overview of the field, reviewing several aTAM results.

Fix an alphabet Σ. Σ^* is the set of finite strings over 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 $G^\mathrm{f}_A = (V, E)$, 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^f_S is connected. A shape Υ is a *tree* if G^f_Υ 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 $G^{\mathrm{f}}_{\mathrm{dom}\,\alpha}$ is connected and $\mathrm{dom}\,\alpha \neq \varnothing$. The *shape* $S_\alpha \subseteq \mathbb{Z}^2$ of α is $\mathrm{dom}\,\alpha$. 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^{b}_α, 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^{b}_α 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 $\alpha \to_1^{\mathcal{T}} \beta$ 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 $\alpha \to_1^{\mathcal{T}} \beta$, $S_\beta \setminus S_\alpha = \{p\}$, and $t = \beta(p)$, we write $\beta = \alpha + (p \mapsto t)$. The $\mathcal{T}$-*frontier* of α is the set

$$\partial^{\mathcal{T}} \alpha = \bigcup_{\alpha \to_1^{\mathcal{T}} \beta} S_\beta \setminus S_\alpha \,,$$

the set of empty locations at which a tile could stably attach to α.

A sequence of $k \in \mathbb{Z}^+ \cup \{\infty\}$ assemblies $\alpha_0, \alpha_1, \ldots$ is a $\mathcal{T}$-*assembly sequence* if, for all $1 \leq i < k$, $\alpha_{i-1} \to_1^{\mathcal{T}} \alpha_i$. We write $\alpha \to^{\mathcal{T}} \beta$, and we say α $\mathcal{T}$-*produces* β (in 0 or more steps) if there is a $\mathcal{T}$-assembly sequence $\alpha_0, \alpha_1, \ldots$ 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 $\sigma \to^{\mathcal{T}} \alpha$, and we write $\mathcal{A}[\mathcal{T}]$ to denote the set of $\mathcal{T}$-producible assemblies. The relation $\to^{\mathcal{T}}$ is a partial order on $\mathcal{A}[\mathcal{T}]$ [26,37]. A $\mathcal{T}$-assembly sequence $\alpha_0, \alpha_1, \ldots$ 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 $\partial^{\mathcal{T}} \alpha = \varnothing$. We write $\mathcal{A}_\Box[\mathcal{T}] \subseteq \mathcal{A}[\mathcal{T}]$ 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 $(\mathcal{A}[\mathcal{T}], \to^{\mathcal{T}})$ is directed; i. e., if for each $\alpha, \beta \in \mathcal{A}[\mathcal{T}]$, there exists $\gamma \in \mathcal{A}[\mathcal{T}]$ such that $\alpha \to^{\mathcal{T}} \gamma$ and $\beta \to^{\mathcal{T}} \gamma$.[5] We say that a TAS $\mathcal{T}$ *strictly self-assembles* a shape $S \subseteq \mathbb{Z}^2$ if, for all $\alpha \in \mathcal{A}_\Box[\mathcal{T}]$, $S_\alpha = S$; i. e., if every terminal assembly produced by $\mathcal{T}$ has shape S. If $\mathcal{T}$ strictly self-assembles some shape S, we say

[2] For $G^{\mathrm{f}}_{S_\alpha} = (V_{S_\alpha}, E_{S_\alpha})$ and $G^{\mathrm{b}}_\alpha = (V_\alpha, E_\alpha)$, G^{b}_α is a spanning subgraph of $G^{\mathrm{f}}_{S_\alpha}$: $V_\alpha = V_{S_\alpha}$ and $E_\alpha \subseteq E_{S_\alpha}$.

[3] Intuitively $\alpha \to_1^{\mathcal{T}} \beta$ 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.

[4] If we had defined the relation $\to^{\mathcal{T}}$ based on only finite assembly sequences, then $\to^{\mathcal{T}}$ would be simply the reflexive, transitive closure $(\to_1^{\mathcal{T}})^*$ of $\to_1^{\mathcal{T}}$. 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.

[5] The following two convenient characterizations of "directed" are routine to verify. $\mathcal{T}$ is directed if and only if $|\mathcal{A}_\Box[\mathcal{T}]| = 1$. $\mathcal{T}$ 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 $\alpha(p) \neq \beta(p)$.

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. If $\mathcal{T}$ strictly self-assembles S and $\mathcal{T}$ is directed, then we say that $\mathcal{T}$ *uniquely self-assembles S*.

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 $\to_1$, $\to$, $\partial\alpha$, *frontier*, *assembly sequence*, *produces*, *producible*, 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 null" single or double glues.[7]

3 Assembly of finite shapes

In this section we study the power of nondeterminism in assembling finite shapes. We first show 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

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

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

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

Using the terminology of the introduction, $\mathrm{C}^{\mathrm{tc}}(S)$ is the minimum number of tile types required to strictly self-assemble S, and $\mathrm{C}^{\mathrm{dtc}}(S)$ is the minimum number of tile types required to unique self-assemble S.

The temperature parameter $\tau = 2$ means that we distinguish two energy levels of bonds, "strong" (strength 2, which can bind on their own) and "weak" (strength 1, which require two cooperating bonds to bind). Higher temperatures mean more discrete energy levels available, which potentially leads to more precise control over tile placement. It is worth mentioning that the temperature potentially affects the tile complexity of a shape. Seki and Okuno [43] considered, for an arbitrary constant $\tau \in \mathbb{Z}^+$, the *directed temperature-at-most-τ tile complexity* of S, denoted by

$$\mathrm{C}^{\mathrm{dtc}}_{\tau}(S) = \min\left\{|T| \;\middle|\; \mathcal{T} = (T, \sigma, \tau') \text{ is a directed TAS, } \tau' \le \tau \text{, and } \mathcal{T} \text{ strictly self-assembles } S\right\}.$$

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

[7] Thus 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.

They proved that for any temperature $\tau \geq 2$, there is a finite shape S with $\mathrm{C}^{\mathrm{dtc}}_{\tau}(S) < \mathrm{C}^{\mathrm{dtc}}_{\tau-1}(S)$. In practice, only temperature-2 self-assembly has been experimentally implemented [7, 20, 38]. Therefore, in this paper we focus on temperature-2 TASs. Our main theorems hold even if we use an arbitrary positive constant temperature $\tau > 2$ in place of 2.

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

$$\textsc{MinTileSet} = \left\{ \langle S, c \rangle \;\middle|\; S \in \mathcal{FS},\, c \in \mathbb{Z}^+, \text{ and } \mathrm{C}^{\mathrm{tc}}(S) \leq c \right\},$$

and the *minimum directed tile set* problem is

$$\textsc{MinDirectedTileSet} = \left\{ \langle S, c \rangle \;\middle|\; S \in \mathcal{FS},\, c \in \mathbb{Z}^+, \text{ and } \mathrm{C}^{\mathrm{dtc}}(S) \leq c \right\}.$$

Adleman, Cheng, Goel, Huang, Kempe, Moisset de Espanés, and Rothemund [4] showed that the problem $\textsc{MinDirectedTileSet}$ is NP-complete. In Section 3.2 we show that $\textsc{MinTileSet}$ is Σ_2^{P}-complete, where $\Sigma_2^{\mathsf{P}} = \mathsf{NP}^{\mathsf{NP}}$. See [6] for a discussion of these complexity classes.

3.1 A finite shape for which nondeterminism reduces tile complexity

Although the main result of Section 3, Theorem 3.2, together with the (widely-believed) assumption that $\mathsf{NP} \neq \Sigma_2^{\mathsf{P}}$ and the fact proven in [4] that $\textsc{MinDirectedTileSet} \in \mathsf{NP}$, implies Theorem 3.1 of this subsection, we prove Theorem 3.1 explicitly in order to illustrate some of the reasoning used in the proof of Theorem 3.2.

Given a shape S (possibly a subshape of a larger shape we wish to self-assemble), we say some tile types *hard-code* S to mean that there are $|S|$ unique tile types, each one specific to a position in S, using double glues between tile types of all adjacent positions in S.

Given a shape S with a subshape $S' \subset S$, we say S' is an *isolated* subshape of S if there is a point $p \in S'$ such that every path from a point in S' to a point in $S \setminus S'$ includes p. In this case, we say p is the *root* of the subshape. If S' is a tree, we say it is an *isolated subtree* of S. We say that an isolated subshape S' of S is *singly-connected* if there is precisely one point in $S \setminus S'$ adjacent to the root of S'.

Theorem 3.1. *There is a finite shape $S \subset \mathbb{Z}^2$ such that* $\mathrm{C}^{\mathrm{tc}}(S) < \mathrm{C}^{\mathrm{dtc}}(S)$.

Proof. As such a shape S, we propose the shape illustrated in Figure 2. The shape S consists of two copies of a subshape S', which is in a labeling box in the figure, and a bridge that connects one of the 6 bottom positions of the grey scaffolds of a copy of S' with the corresponding position of the other copy. The shape S' contains a subshape called the loop L that consists of $L_0, L_1, \ldots, L_h$, $L'_0, L'_1, \ldots, L'_h$, the tile between L_h and L'_h, and the tile between L_0 and L'_0. The height h is left as a variable parameter; increasing h increases the gap between $\mathrm{C}^{\mathrm{tc}}(S)$ and $\mathrm{C}^{\mathrm{dtc}}(S)$.

First we establish that $\mathrm{C}^{\mathrm{tc}}(S) \leq 2h + 30$ (actually with equality, but we only require and hence prove an upper bound). A TAS $\mathcal{T}$ that strictly self-assembles S is designed in the following manner. Its seed is placed somewhere on the connecting bridge of S, and the bridge is hardcoded using 13 tile types. The hardcoding allows $\mathcal{T}$ to put tiles of the same type at the leftmost and rightmost positions of the bridge, to the north of which the two copies of S' can assemble by reusing other $(2h + 16)$ tile types. By having $\mathcal{T}$

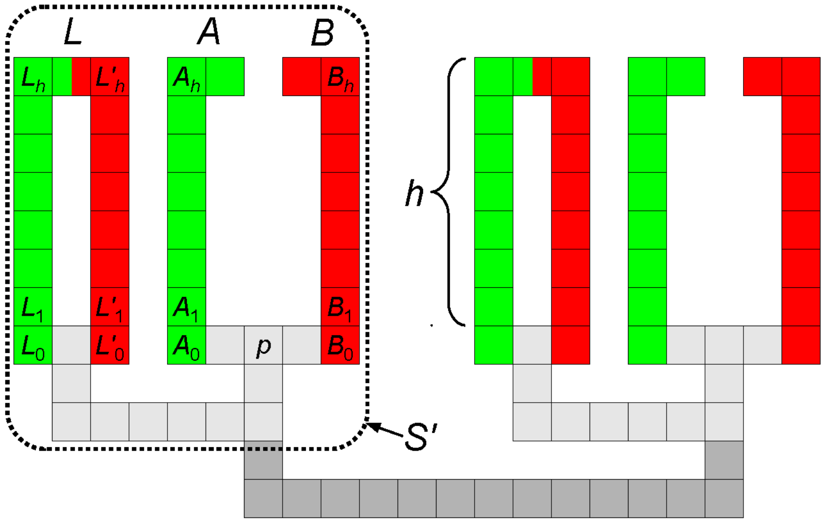

Figure 2: A finite shape S for which $\mathrm{C}^{\mathrm{tc}}(S) < \mathrm{C}^{\mathrm{dtc}}(S)$. Nondeterminism is forced to occur at the two-color top-middle position of the loop L, since any minimal tile set must reuse the tile types from subtrees A and B to create L.

reuse h tile types for $A_0, \ldots, A_h$ and for $L_0, \ldots, L_h$ and other h tile types for $B_0, \ldots, B_h$ and for $L'_0, \ldots, L'_h$, $\mathcal{T}$ can assemble a copy of S' using $2h+16$ tile types. Note that this TAS is not directed because the top-middle position of the loop could receive either a tile from A or a tile from B. Furthermore, these must be different tile types, because the top-right tile type in A must have a double glue on its west but cannot have a double glue on any other side, whereas the top-left tile type of B must have a double glue on its east but not on any other side.

We now show that this nondeterminism is necessary to achieve the minimum tile complexity by showing that $\mathrm{C}^{\mathrm{dtc}}(S) \geq 3h$. Let $\mathcal{T}'$ be a directed TAS that strictly self-assembles S, that is, it admits the unique terminal assembly α with $S_\alpha = S$. Its seed is placed either on one of the two copies of S' or on the connecting bridge, and this means that the other copy of S' grows from the position where the bridge is connected. We will see that the subprocess to assemble the seed-free copy of S' costs $3h$ tile types, which implies $\mathrm{C}^{\mathrm{dtc}}(S) \geq 3h$.

Let S_T be the tree which consists of the subtrees A, B and the three tiles connecting them. In order to assemble the S_T of the seed-free copy of S', $\mathcal{T}'$ needs $2h+7$ tile types due to Theorem 4.3 in [4]. Now we will prove that the loop L of the copy costs $\mathcal{T}'$ h tile types more. Depending on how this loop is assembled, we have two cases to be examined. First we consider the case when the loop is assembled so as for every pair of adjacent tiles on it to be bound via double glue. Then, it is possible for $\mathcal{T}'$ to assemble the left pillar upward as $L_1 \to L_2 \to \cdots \to L_h$. Being doubly-bonded with its right neighbor, the tile that $\mathcal{T}'$ puts on the top-middle position of the loop L should be different from the one on the end of the subtree A. This means that the left pillar cannot reuse the tiles on A. If the left pillar reused any tile type on B instead, then a tile would stick out to the left of the left pillar.

The other case is when some of adjacent tiles on the loop L are not bound via double glue. Note that at most 2 such weak bonds can appear on the loop, and furthermore, they must be incident on a single tile. Thus, we assume without loss of generality that such a weak bond does not occur on the left pillar. If L'_h and the tile on the top-middle of L are bonded by double glue, then all tiles on the positions $L_0, L_1, \ldots, L_h$ as well as the top-middle and L'_h are bounded by double glue and, hence, the process of assembling α can proceed upward from L_0 to L'_h. This process should not reuse any tile type on the subtree A in order to prevent the tile type $\alpha(L'_h)$ from filling the gap between the subtrees A and B. Clearly, it cannot borrow any tile type from the subtree B, either. If the bond is weak, the right pillar must grow upward. Hence, this pillar cannot reuse any tile type on the subtree B because otherwise $\mathcal{T}'$ would produce another terminal assembly in which the tile on the top-middle position of L is bonded with its right neighbor by double glue. As mentioned above, it is almost trivial that the right pillar cannot reuse any tile type on the subtree A. $\square$

3.2 The Minimum Tile Set problem is Σ_2^{P}-complete

The following is the main theorem of this paper.

Theorem 3.2. MINTILESET *is* Σ_2^{P}*-complete.*

Proof. To show that MINTILESET $\in \Sigma_2^{\mathsf{P}}$, define the verification language

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

Clearly $\textsc{MinTileSet}_V \in \mathsf{P}$. MINTILESET $\in \Sigma_2^{\mathsf{P}}$ because $\langle S, c \rangle \in$ MINTILESET if and only if there exists $\mathcal{T} = (T, \sigma, 2)$ with $|T| \le c$ such that for all $\mathcal{T}$-assembly sequences $\vec{\alpha} = (\sigma, \alpha_2, \ldots, \alpha_k)$ of length $k = |S|$, $\langle S, c, \mathcal{T}, \vec{\alpha} \rangle \in \textsc{MinTileSet}_V$, 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 $\exists\forall$CNF-UNSAT is polynomial-time many-one reducible to MINTILESET, where $\exists\forall$CNF-UNSAT is the Σ_2^{P}-complete language [40, 45, 50]

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

We follow a similar strategy to the reduction of 3SAT to MINDIRECTEDTILESET shown in [4]. The 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 [4] 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.[8]

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

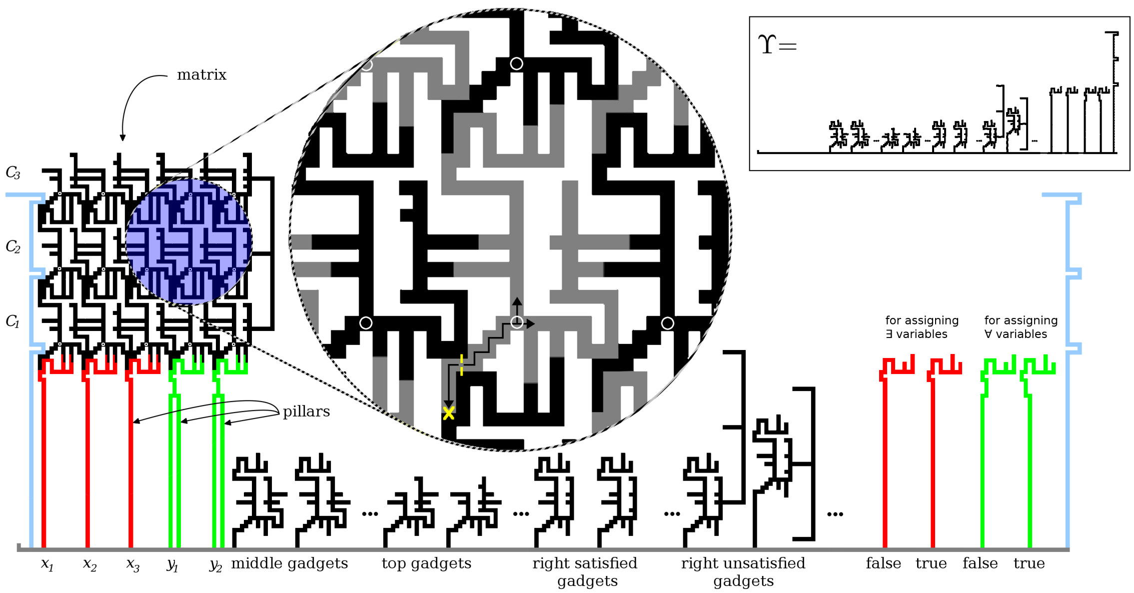

Figure 3: The shape S of the reduction $\langle\varphi\rangle \mapsto \langle S, c\rangle$ showing ∃∀CNF-UNSAT is polynomial-time many-one reducible to MINTILESET. In this example, the quantified negated CNF formula $\varphi = \exists x \forall y \neg\phi(x,y)$ has clauses C_1, C_2 and C_3, ∃-variables x_1, x_2, and x_3, and ∀-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.

Informal description of how the shape S relates to the formula φ Informally, the shape S encodes the formula φ as follows. Each of the gadgets shown in Figure 4 corresponds to a specific variable-clause pair. The "matrix" in Figure 3 is self-assembled by these gadgets, whose purpose is to evaluate the formula clause-by-clause by its own self-assembly. Each row in the matrix of the shape S corresponds to a clause, and each column to a variable. In Figure 3, there are three rows and five columns in the matrix, so a total of fifteen gadgets are used in the matrix. We imagine the matrix assembling one row at a time (although multiple rows could grow in parallel, but for one row to complete, the row below it must complete). The type of gadget at position (i, j) used depends on whether a previous variable (one with index $< i$) satisfied the jth clause, and if not, whether the truth assignment to the ith variable now satisfies the jth clause. For the formula to be satisfied, all clauses must be satisfied, so every gadget in the rightmost column (corresponding to the last variable) must be of type "satisfied," and if so, the black part of the shape to the right of the matrix (appearing like a three-rung ladder in Figure 3) will *not* form. The presence of at least one gadget in the rightmost column of type "unsatisfied" will form the ladder. In

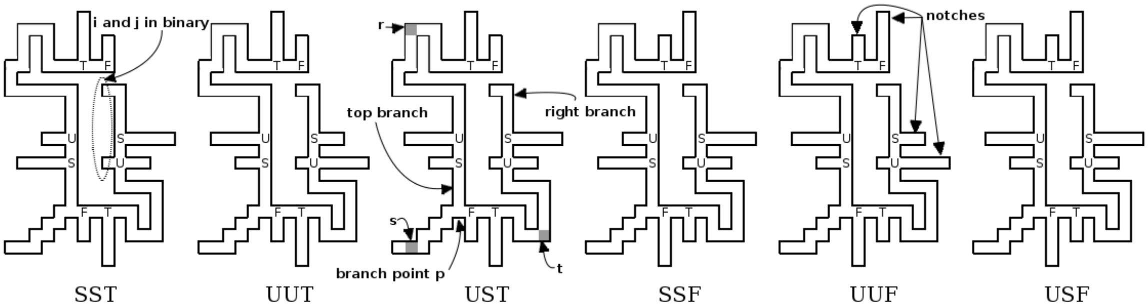

Figure 4: Six main varieties of "information-bearing" tree gadgets used in the reduction. The position (i, j) where the gadget is intended to go in the matrix is encoded in binary. Gadgets intended for the top row are missing the top "T/F" bumps, and gadgets intended for the right column are different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

other words, the S will be strictly self-assembled if and only if the variable assignment chosen (how this is done is described below) does not satisfy ϕ.

The way that variables are assigned to do this evaluation of the formula is as follows. The choice of assignments to the $\exists$-variables is done by design of the tile set, by choosing appropriate glues to make one of each of the two types of red "pillars" from the right side of S grow on the left side of S to start the assembly of the matrix. The choice of assignments to $\forall$-variables is not controlled by the design of the tile set. Rather, to grow the matrix (from a minimal number of tile types), one is forced to choose glues so as to allow both types of green pillars to assemble under the matrix, two for each column corresponding to a $\forall$-variable. Therefore the two green pillars (one representing True and the other False) nondeterministically compete to assign a value to that variable. This is the primary way nondeterminism is used in assembling the shape: all possible assignments to the $\forall$-variables are possible to choose, and if and only if all of them fail to satisfy the formula is the shape S assembled properly (each possible assignment to the $\forall$-variables corresponds to a different terminal assembly, but they all have the shape S if they all make ϕ evaluate to False). It is this ability to "try out" different assignments to the $\forall$-variables, which lead to different assemblies, so long as they all form the same shape, that confers the extra power on non-directed assembly of the shape. The ability manifests in adding an extra $\forall$ quantifier to the Boolean formula, which makes it a Σ_2^{P}-formula.

Formal description of reduction We now describe the reduction more formally.

Suppose that ϕ has k clauses $C_1, \ldots, C_k$ and $\ell = n + m$ input variables $v_1, \ldots, v_\ell$, where $v_1, \ldots, v_n = x_1, \ldots, x_n$ are the $\exists$-variables of φ and $v_{n+1} \ldots, v_\ell = y_1, \ldots, y_m$ 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 $1 \le i \le k$ and $1 \le j \le \ell$, define the following six gadgets:

SST$_{ij}$**:** C_i is satisfied by v_p for some $1 \le p < j$, and v_j is true.

SSF$_{ij}$**:** C_i is satisfied by v_p for some $1 \le p < j$, and v_j is false.

UUT$_{ij}$**:** C_i is unsatisfied by v_p for every $1 \le p \le j$, and v_j is true.

UUF$_{ij}$**:** C_i is unsatisfied by v_p for every $1 \le p \le j$, and v_j is false.

UST$_{ij}$**:** C_i is unsatisfied by v_p for every $1 \le p < j$, C_i is satisfied by v_j, and v_j is true.

USF$_{ij}$**:** C_i is unsatisfied by v_p for every $1 \le p < j$, 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.[9] 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 $1 \le j < \ell$. 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 $1 \le p < j$, 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 $c = \mathrm{C}^{\mathrm{tc}}(\Upsilon)$. We assume that the seed is placed on the rightmost position of the bottom row, for both the shapes Υ and S. At the end of the proof we show how to modify the shapes to enforce this restriction. The steps needed to complete the proof are divided into several lemmas. These lemmas are proven after the current proof. Lemmas 3.3 and 3.6 establish each direction of the claim that φ is true $\iff \mathrm{C}^{\mathrm{tc}}(S) \le c$. 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 $\implies \mathrm{C}^{\mathrm{tc}}(S) \le c$ 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 allow a new gadget encoding the proper information to be placed in the matrix. Therefore the assembly

[9] Strength-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 $\le c$ tile types strictly self-assembles S, then φ must be true.

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 3.4). 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.

To handle the placement of the seed, we use the same trick as in the proof of Theorem 3.1, making two copies of the shape connected by a bridge to enforce that at least one copy does not contain the seed. Define $a \in S$ to be the rightmost point on the bottom row of S. Make two copies of S, place one directly above the other but without touching, and connect the copies by a width-1 "bridge" of length h that connects to each copy of a on a's right side. Denote this new shape by S'. It is routine to show using techniques similar to those in the proof of Theorem 3.1 that any minimal TAS for S' uses $\mathrm{C}^{\mathrm{tc}}(S)+h$ tile types, places the seed in the bridge, uses h tile types to grow the bridge and uses the tile types of a TAS $\mathcal{T}$ that is minimal (subject to the restriction that $\mathcal{T}$ places the seed at a) for S, to assemble each copy of S. Let $c' = c+h$. Our reduction outputs $\langle S', c'\rangle$, rather than $\langle S, c\rangle$. By the arguments above concerning S, we have that φ is true $\iff$ $\mathrm{C}^{\mathrm{tc}}(S') \le c'$, whence $\exists\forall$CNF-UNSAT is polynomial-time many-one reducible to MINTILESET. $\square$

In the following lemmas, φ denotes an arbitrary (true or false) quantified Boolean formula of the form $\varphi = \exists x \forall y \neg\phi(x,y)$, where ϕ is an unquantified CNF formula with $n+m$ input bits $x = x_1,\dots,x_n$ and $y = y_1,\dots,y_m$. $\Upsilon, S \in \mathcal{FS}(\mathbb{Z}^2)$ refer to the tree and shape constructed from φ as in the proof of Theorem 3.2, and $c = \mathrm{C}^{\mathrm{tc}}(\Upsilon)$.

Lemma 3.3. *If φ is true, then $\mathrm{C}^{\mathrm{tc}}(S) \le c$.*

Proof. Let $\mathcal{T}_\Upsilon = (T,\sigma,2)$ be a minimal TAS that strictly self-assembles Υ with seed placed at position a, the rightmost point of the bottom row of Υ, and let $\alpha \in \mathcal{A}_\Box[\mathcal{T}_\Upsilon]$ be the unique terminal producible assembly of $\mathcal{T}_\Upsilon$, such that $S_\alpha = \Upsilon$. Theorem 4.3 of [4] shows that if $\mathcal{T}_\Upsilon$ is a minimal TAS for Υ, then $\mathcal{T}_\Upsilon$ puts the same tile type in two positions $p_1, p_2 \in \Upsilon$ if and only if the subtrees of Υ rooted at p_1 and p_2 (with the seed location considered the root of Υ) are isomorphic and "identically entered" (meaning both of them have their parent in the same direction). This theorem is stated for directed TAS's but it is easy to show that any minimal TAS for a tree must be directed. Given α and $t \in T$, define t to be *singular in* α if it appears exactly once in α. Thus, all the positions at the bottom row of Υ will receive tile types that are singular in α.

Since φ is true, there is an assignment f to variables $x_1,x_2,\dots,x_n$ such that any assignment to $y_1,y_2,\dots,y_m$ makes $\phi(x,y)$ false. Since all the tile types in the bottom row are unique, we can change the north glues of the tiles at the base of the clause-variable matrix, without ruining the rest of the shape. We change the north glues so that the blue pillar grows as the leftmost pillar and for each variable x_i we grow the red true/false pillar depending on $f(x_i)$ being true or false. For the positions corresponding to y variables, we change the north glues so that both true and false green pillars can grow.

We will also change a number of null glues into single glues in the following way: the set of labels with strength one will be $\mathtt{T}$, $\mathtt{F}$, $\mathtt{S}_{i,j}$, and $\mathtt{U}_{i,j}$, where $1 \le i \le k$ and $1 \le j \le \ell$. For each gadget G, let p be the position of its branch point in Υ, as shown in Figure 4. Then, we change the null glue of the south side of $\alpha(s)$ to a single glue with label T if G is of type *SST*, *UUT*, or *UST*; otherwise, we change the south glue to a single glue with label F. Also, we change the north glue of $\alpha(s)$ to a single glue with label $\mathtt{S}_{i,j}$ if G is a gadget for the ith clause and jth variable and is of type *SST* or *SSF*; otherwise, if G is of type *UUT*, *UUF*, *UST*, or *USF*, the north glue of $\alpha(s)$ will be a single glue with label $\mathtt{U}_{i,j}$. Note that, the tile type $\alpha(s)$ is singular in α, due to the fact that its rooted subtree contains the encoding about the gadget type, clause number, and variable number, and hence, is not isomorphic to any other subtree.

The north glue of the tile type $\alpha(r)$ will be changed to a strength one glue with label T if G is of type *SST*, *UUT*, or *UST*; otherwise, its label will be F. The south glue of the tile type $\alpha(t)$ will be changed to a strength one glue with label $\mathtt{S}_{i,j+1}$ if G is for the ith clause and jth variable and is of type *SST*, *SSF*, *UST*, or *USF*; otherwise, its label will be $\mathtt{U}_{i,j+1}$.

Applying the above-mentioned changes will give us a TAS $\mathcal{T}_S$ with the same tile complexity as $\mathcal{T}_\Upsilon$. In $\mathcal{T}_S$, a gadget can grow at the cell of the matrix corresponding to clause i and variable j using cooperation of single glues of the bottom and left gadget if and only if its notches match the notches of the bottom and left gadget. And, the notches of gadgets are designed in a way that they can be put together to assemble S if and only if the truth assignment to x and y variables (presented as pillar notches at the first row of the matrix) make $\phi(x,y)$ false. Intuitively, the gadgets grow in the matrix so as to "evaluate $\phi(x,y)$" on inputs x and y encoded in the pillars, with the pillars encoding y nondeterministically choosing values for each of the y_i. If we choose the proper assignment f to x such that, for all assignments to y (corresponding to different terminal assemblies), $\phi(x,y)$ is false (such an assignment f exists since $\varphi = \exists x \forall y \neg\phi(x,y)$ is true), then all of these assemblies will have at least one unsatisfied right gadget in the rightmost column of the matrix, and the assembly will have shape S. Therefore $\mathcal{T}_S$ strictly self-assembles S. □

From here until the end of the section, 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. Also, B 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:

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

Let $G^* = \bigcap_{G \in \mathcal{G}} G$, where each element of $\mathcal{G}$ is a set of tile positions of a gadget created in the proof of Theorem 3.2 (Figure 4) translated so that the branch tile is at the origin (so, $\mathcal{G}$ has at most $k \times \ell \times 6$ elements).

Lemma 3.4. *If $|T_S| \le c = \mathrm{C}^{\mathrm{tc}}(\Upsilon)$, 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 $\mathcal{T}_\Upsilon$ strictly self-assembles Υ.*

Proof. Let $\alpha \in \mathcal{A}_\square[\mathcal{T}_S]$. Let the subassembly of α consisting of the gray row underneath the matrix, between the leftmost and rightmost pillars in Figure 3, be denoted by R. It suffices to show that all tile types in R appear precisely once in $\alpha \upharpoonright \Upsilon$ (α restricted to Υ), and further that they are all bound together by double glues. This will establish that, by adjusting double glues on the north of tiles beneath the pillars to be null glues, all of Υ can grow from the tiles without growing any of the pillars or matrix of $S \setminus \Upsilon$, and without affecting the assembly of the rest of $\alpha \upharpoonright \Upsilon$. Since we only change double glues to null glues, it is not possible to introduce new double glues adjacent to empty space that were not already present, so nothing will grow outside of Υ.

First we show that every adjacent pair of tiles in R is bound by a double glue. For the sake of contradiction, let $p_1 \in \Upsilon$ and $p_2 \in \Upsilon$ be the closest positions to the seed that are adjacent to each other but $\alpha(p_1)$ and $\alpha(p_2)$ do not have double glue between them. Then, they are on the bottom row below the clause-variable matrix in S. Thus, there must be a pillar growing down from the clause-variable matrix; consider the pillar that grows down in α earlier than the others. This pillar cannot reuse any tile type that is used in positions to the right of p_1 in Υ; otherwise, an undesirable part of Υ can grow to the left of the downward pillar. Therefore, the number of tile types in $\mathcal{T}_S$ is at least $\mathrm{C}^{\mathrm{tc}}(\Upsilon) - x + y$, where x is the horizontal width of the clause-variable matrix and y is the height of the pillars. This is a contradiction to the assumption that $\mathcal{T}_S$ uses at most $\mathrm{C}^{\mathrm{tc}}(\Upsilon)$ tile types, since we set $y > x$ in our construction. Thus every adjacent pair of tiles in R is bound by a double glue.

We now argue that each tile type in R appears exactly once in $\alpha \upharpoonright \Upsilon$. Since the portion of Υ outside of R is acyclic (and since we just showed tiles within R are double-bonded to each other), then all of $\alpha \upharpoonright \Upsilon$ must be entirely double-bonded. Since Υ is a tree shape, there is no possibility of the presence of one tile in Υ "blocking" the growth of another in Υ, since this would introduce a cycle in the shape being assembled and it would not be a tree. Therefore, for any two positions p_1 and p_2 in Υ such that $\alpha(p_1) = \alpha(p_2)$, any subtree of Υ rooted at p_1 (with respect to the seed location being the root of the whole tree) must also appear as a subtree of p_2. But every subtree within Υ of every position p_1 in R (each such subtree being simply the portion of R to the left of p_1, plus the gray row to the left of that, plus the single "notch" position just above the leftmost position on the row) does not appear anywhere else in Υ. This can be verified by inspection of Figure 3. Therefore the tile types in R do not appear anywhere else in $\alpha \upharpoonright \Upsilon$. □

The following lemma states the "inductive step" of the proof of Lemma 3.6. 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 3.5. *Suppose $\mathcal{T}_S$ has at most $c = \mathrm{C}^{\mathrm{tc}}(\Upsilon)$ 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 rooted at tiles in $S_\alpha \cap I$ are present in α, and*
5. *if there exists $p \in I$ such that $p \notin S_\alpha$, then $S_\alpha \cap (p + G^*) = \varnothing$.*

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

Proof. α cannot be a terminal assembly, since $\mathcal{T}_S$ strictly self-assembles $S \neq S_\alpha$.

Since double glues cannot be added to gadget tile types without ruining the tree shape portion of S, α must grow by cooperation of single glues. This cooperation can happen only at $F-(4,3)$, where

$$F = \{p \in I \setminus S_\alpha \mid (S_\alpha \cap I) \cup \{p\} \text{ is a staircase}\} .$$

In other words, $F-(4,3)$ is the set of points marked s in Figure 4, which are adjacent to points marked r and t in neighboring gadgets. Let β' be a minimal assembly producible from α such that $S_{\beta'} \cap F$ is not empty. Let $\{p\} = S_{\beta'} \cap F$. All paths from S_α to p in β' must pass through $s = p-(4,3)$, due to the minimality of β'. Even if the tile type that goes in s is not taken from any gadgets, the tile type that eventually goes to $s+(1,1)$ must be chosen from a gadget, since it should be able to grown the zig-zag shape only by double glues, and the zig-zag shape is used only in one of the gadgets. Thus, $\beta'(p)$ is a branch tile type.

Since $p \in S_{\beta'} \cap I$ and $p \notin S_\alpha \cap I$, $S_{\beta'} \cap I$ has cardinality at least $m+1$. Let β be the minimal assembly producible from β' in which the right and top branches of p are tiled. $\beta(p)$ and $\beta(q)$ for all q in the branches of p must be tile types from the correct gadget to ensure consistency with the notches of neighboring gadgets and consistency with the (row,column) identifier notches at the top of the right branch.

Due to the minimality of β, it satisfies condition 5. □

Lemma 3.6. *If $\mathcal{T}_S$ has at most c tile types, then φ is true.*

Proof. First we show that B is $\mathcal{T}_S$-producible. According to Lemma 3.4, by changing a number of double glues in $\mathcal{T}_S$ to null glues, we can obtain a TAS $\mathcal{T}_\Upsilon$ that strictly self-assembles Υ. So, Υ is $\mathcal{T}_S$-producible. Moreover, as can be checked in the proof of Lemma 3.4, the null glues in $\mathcal{T}_\Upsilon$ that are double glues in $\mathcal{T}_S$ are the north glues of the tile types that appear at the base of pillars beneath the gadget matrix. Also, all the pillars must grow from the bottom row to the matrix, and not downward, because growing a pillar downward requires adding a double glue to a tile type in the matrix area, which will also ruin Υ. Thus, B, which is Υ together with the pillars beneath the matrix, is $\mathcal{T}_S$-producible. This establishes the base case.

Let $f(x_i)$ be true if the red true pillar is used to grow the pillar corresponding to x_i and be false if the red false pillar is used. Using Lemma 3.5 for the inductive case, we conclude that there is an assembly α such that $B \cup I \subseteq S_\alpha$ and valid gadgets are/can be used to fill the matrix part of α. By our construction of gadgets, this implies that the truth assignment f to x makes $\phi(x,y)$ false for every value of y. Thus φ is true. □

4 Assembly of infinite shapes

In this section we study the power of nondeterminism in assembling infinite shapes. The following theorem, an infinitary analog of Theorem 3.1, is the main result of Section 4.

Theorem 4.1. *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.*

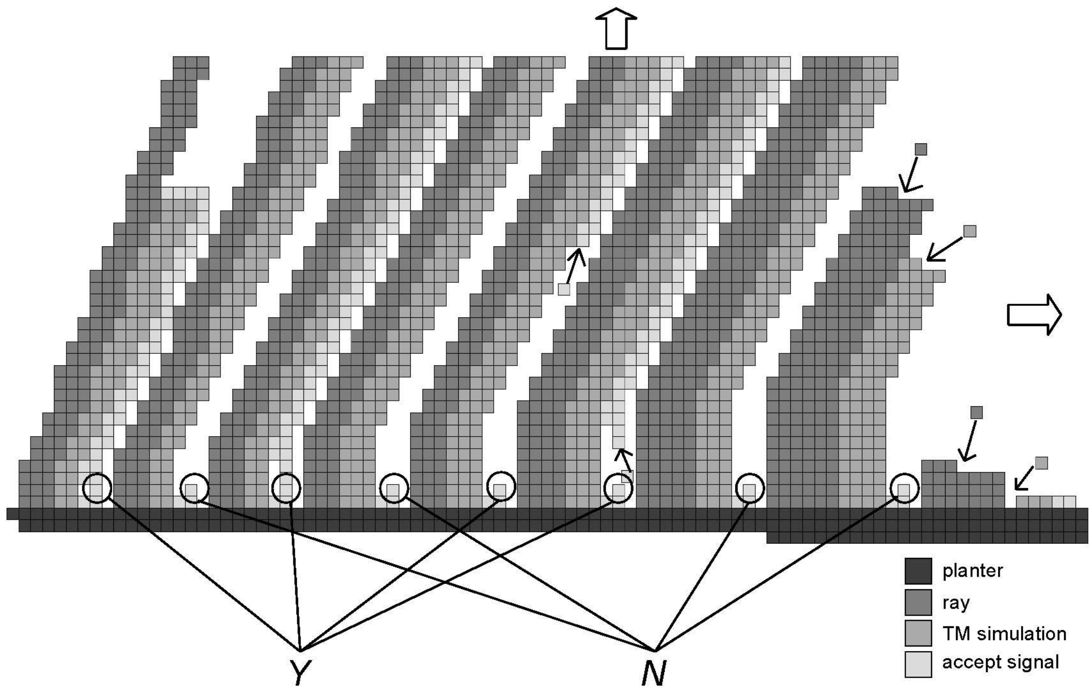

Figure 5: A portion of an infinite shape S that strictly self-assembles, but not by any directed TAS. The n^{th} ray simulates a Turing machine M on input n, and a vertical line is present under that ray if and only if M accepts n. Y and N are points at these positions representing "yes" and "no" instances 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.

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 $L = L(M) = \{n \in \mathbb{N} \mid M \text{ accepts } n\}$. 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 5. The TAS that strictly self-assembles S is based on the main construction of Lathrop, Lutz, Patitz, and Summers [25]. 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 $f : \mathbb{N} \to \mathbb{N}$, 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 5 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 $n \to \infty$. 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.[10]

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 [25]. 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 uniquely 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 5. Let $T_Y = \alpha(Y)$ and $T_N = \alpha(N)$ be the set of tile types that appear at "yes" and "no" 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 $T_Y \cap T_N = \varnothing$. 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

[10] [25] 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.

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 $T_Y \cap T_N = \varnothing$.

For all $n \in \mathbb{N}$, let $p_n = (f(n), 0)$. Since $T_Y \cap T_N = \varnothing$, for all $n > n_0$,

$$n \in L \iff p_n \in Y \iff \alpha(p_n) \in T_Y\,, \quad \text{and} \quad n \notin L \iff p_n \in N \iff \alpha(p_n) \in T_N\,.$$

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. □

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 [36] available here:

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 or the TAS that strictly self-assembles the shape. Furthermore, the bulk of the intellectual effort in the proof of Theorem 4.1 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 and help to convince oneself that the shape really can be strictly self-assembled and to directly observe how the TAS accomplishes this task.

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 $\mathsf{NP} = \Sigma_2^{\mathsf{P}}$.

There are some interesting questions that remain open:

1. What is the fastest growing function $f : \mathbb{N} \to \mathbb{N}$ 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* $\mathrm{C}^{\mathrm{tc}}(S) \leq n$ *and* $\mathrm{C}^{\mathrm{dtc}}(S) \geq f(n)$"? The proof of Theorem 3.1 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 4.1 establishes such a statement for *all* functions $f : \mathbb{N} \to \mathbb{N}$ 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?

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 $\alpha \in \mathcal{A}_{\Box}[\mathcal{T}]$, $\alpha^{-1}(B) = S$. 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 [24, 28, 33–35, 51] or placement of guides for walking molecular robots [29]; see [25, 26] for more discussion of the theoretical issues of weak self-assembly. It remains open to prove or disprove analogs of Theorems 4.1 and 3.1, 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 4.1 and 3.1, 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 $\mathbb{Z}^2$.

[11] For example, $\mathrm{C}^{\mathrm{tc}}(S) \approx \mathrm{C}^{\mathrm{dtc}}(S) = O(\log n / \log\log n)$ for S an $n \times k$ rectangle with $n \geq k \geq \log n / \log\log n$, but $\mathrm{C}^{\mathrm{tc}}(S)$ and $\mathrm{C}^{\mathrm{dtc}}(S)$ 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.

Define

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

In contrast to the case for strict self-assembly, it can be shown that MINWEAKTILESET is undecidable. This follows from a "Berry's paradox" argument, similar to the one used to show that Kolmogorov complexity is uncomputable, together with the fact that arbitrary algorithms may be simulated in a tile assembly system. Briefly, assuming this quantity is computable, define a Turing machine M that on input $c \in \mathbb{Z}^+$ enumerates finite sets of points lying entirely on the positive x-axis until a set $S(c)$ is found whose "weak self-assembly tile complexity" exceeds c. Then for each $c \in \mathbb{Z}^+$ define a tile system $\mathcal{T}$ that simulates $M(c)$ in the second quadrant, using its output to place black tiles precisely on points in $S(c)$. Since $\mathcal{T}$ requires only $\log c + O(1)$ tile types, for sufficiently large c this contradicts the tile complexity of $S(c)$. It is not difficult to show that MINWEAKTILESET $\in \Pi^0_2$. Is it complete for Π^0_2?

7. MINDIRECTEDWEAKTILESET is defined similarly to MINWEAKTILESET but also requiring $\mathcal{T}$ to be directed. It is not difficult to show that MINDIRECTEDWEAKTILESET $\in \Delta^0_2$. Can be be shown that MINDIRECTEDWEAKTILESET $\notin \Sigma^0_1 \cup \Pi^0_1$?

8. The previous two problems have "bounded" variants:

$$\textsc{MinBddWeakTileSet} = \left\{ \langle S, c, 0^n\rangle \;\middle|\; \begin{array}{l} S \in \mathcal{F}(\mathbb{Z}^2),\ c \in \mathbb{Z}^+,\ \text{and } (\exists \mathcal{T} = (T,\sigma,2))\ |T| \le c \\ \text{and } \mathcal{T} \text{ weakly self-assembles } S \text{ and } (\forall \alpha \in \mathcal{A}_\Box[\mathcal{T}]) \\ S_\alpha \subseteq \{0,\ldots,n-1\}^2 \end{array} \right\},$$

and define MINBDDDIRECTEDWEAKTILESET similarly for the directed case. It is easy to show that MINBDDWEAKTILESET $\in \Sigma^{\mathsf{P}}_2$ and MINBDDDIRECTEDWEAKTILESET $\in \mathsf{NP}$ using the same techniques used for strict self-assembly. Can it be shown that they are complete for these classes? This seems more difficult than the case of strict self-assembly, since the "bounding box" is now constrained to be a square, so that it is no longer possible to use the technique of forcing certain subshapes to be trees (whose tilings by minimal tile sets are well-characterized).

9. In [4] 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 [4] 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 [5,21,27]. 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 [18] implies that it cannot be coNP-hard (nor NP-hard by Mahaney's Theorem [30]) unless $\mathsf{P} = \mathsf{NP}$.

[12]More precisely, if one takes a bit of care in encoding the problem, then it can be assumed sparse. Assume that each square S

Acknowledgement The authors are grateful to anonymous referees for their detailed and useful comments.

References

[1] ZACHARY ABEL, NADIA BENBERNOU, MIRELA DAMIAN, ERIK D. DEMAINE, MARTIN L. DEMAINE, ROBIN Y. FLATLAND, SCOTT D. KOMINERS, AND ROBERT T. SCHWELLER: Shape replication through self-assembly and RNase enzymes. In *Proc. 21st Ann. ACM-SIAM Symp. on Discrete Algorithms (SODA'10)*, pp. 1045–1064. ACM Press, 2010. [ACM:1873601.1873686] 2

[2] LEONARD M. ADLEMAN: Towards a mathematical theory of self-assembly. Technical report, University of Southern California, 2000. 4

[3] LEONARD M. ADLEMAN, QI CHENG, ASHISH GOEL, AND MING-DEH A. HUANG: Running time and program size for self-assembled squares. In *Proc. 33rd STOC*, pp. 740–748. ACM Press, 2001. [doi:10.1145/380752.380881] 2, 23

[4] LEONARD M. ADLEMAN, QI CHENG, ASHISH GOEL, MING-DEH A. HUANG, DAVID KEMPE, PABLO MOISSET DE ESPANÉS, AND PAUL W. K. ROTHEMUND: Combinatorial optimization problems in self-assembly. In *Proc. 34th STOC*, pp. 23–32. ACM Press, 2002. [doi:10.1145/509907.509913] 4, 5, 8, 9, 10, 14, 22

[5] 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 J. Comput.*, 34(6):1493–1515, 2005. Preliminary version in SODA'04. [doi:10.1137/S0097539704445202] 2, 4, 5, 22

[6] SANJEEV ARORA AND BOAZ BARAK: *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009. 8

[7] ROBERT D. BARISH, REBECCA SCHULMAN, PAUL W. K. ROTHEMUND, AND ERIK WINFREE: An information-bearing seed for nucleating algorithmic self-assembly. *Proc. Nat. Acad. Sci.*, 106(15):6054–6059, 2009. [doi:10.1073/pnas.0808736106] 2, 8

[8] FLORENT BECKER, IVAN RAPAPORT, AND ÉRIC RÉMILA: Self-assemblying classes of shapes with a minimum number of tiles, and in optimal time. In *26th Internat. Conf. Foundations of Software Technology and Theoretical Computer Science (FSTTCS'06)*, pp. 45–56, 2006. [doi:10.1007/11944836_7] 2

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 $\equiv^{\mathsf{P}}_{\mathsf{m}}$-equivalent to the minimum tile set problem for squares is

$$\left\{ \langle S,c \rangle \in \text{MINTILESET} \;\middle|\; S \text{ is an } n\times n \text{ square with lower-left corner } (0,0)\text{, and } 1 \le c \le n^2 \right\}.$$

In fact we can even require $c \le O(\log n/\log\log n)$ by a result of [3], but the trivial upper bound of n^2 suffices to obtain sparseness.

[9] NATHANIEL BRYANS, EHSAN CHINIFOROOSHAN, DAVID DOTY, LILA KARI, AND SHINNOSUKE SEKI: The power of nondeterminism in self-assembly. In *Proc. 22nd Ann. ACM-SIAM Symp. on Discrete Algorithms (SODA'11)*, pp. 590–602. ACM Press, 2011. [ACM:2133082] 1

[10] HARISH CHANDRAN, NIKHIL GOPALKRISHNAN, AND JOHN H. REIF: Tile complexity of linear assemblies. *SIAM J. Comput.*, 41(4):1051–1073, 2012. Preliminary version in ICALP'09. [doi:10.1137/110822487] 2

[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. Preliminary version in DNA'07. [doi:10.1007/s11047-008-9073-0] 2

[12] ERIK D. DEMAINE, SARAH EISENSTAT, MASHHOOD ISHAQUE, AND ANDREW WINSLOW: One-dimensional staged self-assembly. In *17th Internat. Conf. DNA Computing and Molecular Programming (DNA'11)*, pp. 100–114, 2011. [doi:10.1007/978-3-642-23638-9_10] 2

[13] ERIK D. DEMAINE, MATTHEW J. PATITZ, ROBERT T. SCHWELLER, AND SCOTT M. SUMMERS: Self-assembly of arbitrary shapes using RNAse enzymes: Meeting the Kolmogorov bound with small scale factor. In *Proc. 28th Symp. Theoretical Aspects of Comp. Sci. (STACS'11)*, pp. 201–212, 2011. [doi:10.4230/LIPIcs.STACS.2011.201] 2

[14] DAVID DOTY: Randomized self-assembly for exact shapes. *SIAM J. Comput.*, 39(8):3521–3552, 2010. Preliminary version in FOCS'09. [doi:10.1137/090779152] 2

[15] DAVID DOTY: Theory of algorithmic self-assembly. *Comm. ACM*, 55(12):78–88, 2012. [doi:10.1145/2380656.2380675] 5

[16] DAVID DOTY, MATTHEW J. PATITZ, AND SCOTT M. SUMMERS: Limitations of self-assembly at temperature 1. *Theoret. Comput. Sci.*, 412(1-2):145–158, 2011. Preliminary version in DNA'09. [doi:10.1016/j.tcs.2010.08.023] 4

[17] CONSTANTINE EVANS: Personal communication. 2

[18] STEVEN FORTUNE: A note on sparse complete sets. *SIAM J. Comput.*, 8(3):431–433, 1979. [doi:10.1137/0208034] 22

[19] BIN FU, MATTHEW J. PATITZ, ROBERT T. SCHWELLER, AND ROBERT SHELINE: Self-assembly with geometric tiles. In *Proc. 39th Internat. Colloq. on Automata, Languages and Programming (ICALP'12)*, pp. 714–725, 2012. [doi:10.1007/978-3-642-31594-7_60] 2

[20] KENICHI FUJIBAYASHI, RIZAL HARIADI, SUNG HA PARK, ERIK WINFREE, AND SATOSHI MURATA: Toward reliable algorithmic self-assembly of DNA tiles: A fixed-width cellular automaton pattern. *Nano Letters*, 8(7):1791–1797, 2008. [doi:10.1021/nl0722830] 2, 8

[21] MICHAEL R. GAREY AND DAVID S. JOHNSON: *Computers and Intractability*. W. H. Freeman, New York, 1979. 22

[22] MING-YANG KAO AND ROBERT T. SCHWELLER: Reducing tile complexity for self-assembly through temperature programming. In *Proc. 17th Ann. ACM-SIAM Symp. on Discrete Algorithms (SODA'06)*, pp. 571–580. ACM Press, 2006. [doi:10.1145/1109557.1109620] 2

[23] MING-YANG KAO AND ROBERT T. SCHWELLER: Randomized self-assembly for approximate shapes. In *Proc. 35th Internat. Colloq. on Automata, Languages and Programming (ICALP'08)*, pp. 370–384. Springer, 2008. [doi:10.1007/978-3-540-70575-8_31] 2

[24] 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. ROTHEMUND, AND GREGORY M. WALLRAFF: Placement and orientation of individual DNA shapes on lithographically patterned surfaces. *Nature Nanotechnology*, 3(9):557–561, 2009. [doi:10.1038/nnano.2009.220] 21

[25] JAMES I. LATHROP, JACK H. LUTZ, MATTHEW J. PATITZ, AND SCOTT M. SUMMERS: Computability and complexity in self-assembly. *Theory of Computing Systems*, 48(3):617–647, 2011. Preliminary version in CiE'08. [doi:10.1007/s00224-010-9252-0] 18, 19, 21

[26] JAMES I. LATHROP, JACK H. LUTZ, AND SCOTT M. SUMMERS: Strict self-assembly of discrete Sierpinski triangles. *Theoret. Comput. Sci.*, 410(4-5):384–405, 2009. Preliminary version in CiE'07. [doi:10.1016/j.tcs.2008.09.062] 6, 21

[27] HARRY R. LEWIS AND CHRISTOS H. PAPADIMITRIOU: *Elements of the Theory of Computation*. Prentice-Hall, Inc., Upper Saddle River, New Jersey, 1998. 22

[28] 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. *Proc. Nat. Acad. Sci.*, 101(3):717–722, 2004. [doi:10.1073/pnas.0305860101] 21

[29] KYLE LUND, ANTHONY T. MANZO, NADINE DABBY, NICOLE MICHOLOTTI, ALEXANDER JOHNSON-BUCK, JEANETTER NANGREAVE, STEVEN TAYLOR, RENJUN PEI, MILAN N. STOJANOVIC, NILS G. WALTER, ERIK WINFREE, AND HAO YAN: Molecular robots guided by prescriptive landscapes. *Nature*, 465(7295):206–210, 2010. [doi:10.1038/nature09012] 21

[30] STEPHEN R. MAHANEY: Sparse complete sets of NP: Solution of a conjecture of Berman and Hartmanis. *J. Comput. System Sci.*, 25(2):130–143, 1982. Preliminary version in FOCS'80. [doi:10.1016/0022-0000(82)90002-2] 22

[31] JÁN MAŇUCH, LADISLAV STACHO, AND CHRISTINE STOLL: Two lower bounds for self-assemblies at temperature 1. *J. Computational Biology*, 17(6):841–852, 2010. Preliminary versions in SAC'09 and ICBBE'09. [doi:10.1089/cmb.2009.0067] 4

[32] JÁN MAŇUCH, LADISLAV STACHO, AND CHRISTINE STOLL: Step-wise tile assembly with a constant number of tile types. *Natural Computing*, 11(3):535–550, 2012. Preliminary version in ISAAC'09. [doi:10.1007/s11047-012-9321-1] 2

[33] HAREEM T. MAUNE, SI-PING HAN, ROBERT D. BARISH, MARC BOCKRATH, WILLIAM A. GODDARD III, PAUL W. K. ROTHEMUND, AND ERIK WINFREE: Self-assembly of carbon nanotubes into two-dimensional geometries using DNA origami templates. *Nature Nanotechnology*, 5(1):61–66, 2010. [doi:10.1038/nnano.2009.311] 21

[34] 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(5):735–739, 2006. [doi:10.1002/anie.200503797] 21

[35] SUNG HA PARK, HAO YAN, JOHN H. REIF, THOMAS H. LABEAN, AND GLEB FINKELSTEIN: Electronic Nanostructures Templated on Self-Assembled DNA Scaffolds. *Nanotechnology*, 15(10):S525–S527, 2004. [doi:10.1088/0957-4484/15/10/005] 21

[36] MATTHEW J. PATITZ: Simulation of self-assembly in the abstract tile assembly model with ISU TAS. In *6th Ann. Conf. Foundations of Nanoscience (FNANO'09)*, pp. 209–219. Sciencetechnica, 2009. [arXiv:1101.5151] 20

[37] PAUL W. K. ROTHEMUND: *Theory and Experiments in Algorithmic Self-Assembly*. Ph. D. thesis, University of Southern California, 2001. 6

[38] PAUL W. K. ROTHEMUND, NICK PAPADAKIS, AND ERIK WINFREE: Algorithmic self-assembly of DNA Sierpinski triangles. *PLoS Biology*, 2(12):2041–2053, 2004. [doi:10.1371/journal.pbio.0020424] 2, 8

[39] PAUL W. K. ROTHEMUND AND ERIK WINFREE: The program-size complexity of self-assembled squares (extended abstract). In *Proc. 32nd STOC*, pp. 459–468. ACM Press, 2000. [doi:10.1145/335305.335358] 2, 5

[40] 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, 2002. [doi:10.1145/582475.582484] 10

[41] ROBERT T. SCHWELLER: Personal communication, 2011. `http://faculty.utpa.edu/rtschweller/papers/TheGap.pdf`. 4

[42] NADRIAN C. SEEMAN: Nucleic-acid junctions and lattices. *Journal of Theoretical Biology*, 99(2):237–247, 1982. [doi:10.1016/0022-5193(82)90002-9] 2

[43] SHINNOSUKE SEKI AND YASUSHI OKUNO: On the behavior of tile assembly system at high temperatures. In *8th Conf. Computability in Europe, (CiE'12)*, pp. 549–559. Springer, 2012. [doi:10.1007/978-3-642-30870-3_55] 7

[44] DAVID SOLOVEICHIK AND ERIK WINFREE: Complexity of self-assembled shapes. *SIAM J. Comput.*, 36(6):1544–1569, 2007. Preliminary version in DNA'04. [doi:10.1137/S0097539704446712] 2, 4

[45] LARRY J. STOCKMEYER: The polynomial-time hierarchy. *Theoret. Comput. Sci.*, 3(1):1–22, 1976. [doi:10.1016/0304-3975(76)90061-X] 10

[46] SCOTT M. SUMMERS: Reducing tile complexity for the self-assembly of scaled shapes through temperature programming. *Algorithmica*, 63(1-2):117–136, 2012. [doi:10.1007/s00453-011-9522-5] 2

[47] HAO WANG: Proving theorems by pattern recognition – II. *The Bell System Technical Journal*, XL(1):1–41, 1961. 2

[48] HAO WANG: Dominoes and the AEA case of the decision problem. In *Proc. Symp. Mathem. Theory of Automata (MTA'62)*, pp. 23–55. Polytechnic Press of Polytechnic Inst. of Brooklyn, Brooklyn, N.Y., 1963. 2

[49] ERIK WINFREE: *Algorithmic Self-Assembly of DNA*. Ph. D. thesis, California Institute of Technology, 1998. 2, 5

[50] CELIA WRATHALL: Complete sets and the polynomial-time hierarchy. *Theoret. Comput. Sci.*, 3(1):23–33, 1976. [doi:10.1016/0304-3975(76)90062-1] 10

[51] 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):1882–1884, 2003. [doi:10.1126/science.1089389] 21

AUTHORS

Nathaniel Bryans
Software Development Engineer in Test
Microsoft, Inc.
Redmond, Washington, USA
nathbry@microsoft.com

Ehsan Chiniforooshan
Software Engineer
Google, Inc.
Waterloo, Ontario, Canada
chiniforooshan@alumni.uwaterloo.ca

David Doty
Computing Innovation Fellow and Postdoctoral Scholar
Computing and Mathematical Sciences
California Institute of Technology
Pasadena, California, USA
ddoty@caltech.edu
http://www.dna.caltech.edu/~ddoty/

Lila Kari
Professor, Computer Science
University of Western Ontario
London, Ontario, Canada
lila@csd.uwo.ca
http://www.csd.uwo.ca/~lila/

Shinnosuke Seki
Postdoctoral Scholar
Information and Computer Science
Aalto University
Helsinki, Finland
shinnosuke.seki@aalto.fi
http://users.ics.aalto.fi/sseki/

ABOUT THE AUTHORS

NATHANIEL BRYANS received his B. Sc. in Bioinformatics at The University of Western Ontario in 2012. He currently works at Microsoft as a Software Development Engineer in Test. In his free time, he enjoys hiking, skiing, and improving his minesweeper score.

EHSAN CHINIFOROOSHAN received his M. Sc. from Sharif University of Technology, advised by Rouzbeh Tusserkani, and his Ph. D. from the University of Waterloo under the supervision of Naomi Nishimura. He has worked on problems in Combinatorics, Graph Theory, Data Structures, and Self-Assembly, is currently a Software Engineer at Google, Kitchener-Waterloo, and likes to bike and play table tennis and Go.

DAVID DOTY received his Ph. D. in Computer Science at Iowa State University in 2009; his advisors were Jack Lutz and Jim Lathrop. His thesis focused on applying the theory of computation to problems in molecular self-assembly. He lives in Pasadena, CA, where he works at Caltech and continues to prove theorems related to molecular computation and conduct physical experiments implementing molecular computation. When he is not proving theorems (and sometimes when he is), he can be found roaming the San Gabriel mountains, with or without a snowboard, and sometimes surfing, cooking, or eating.

LILA KARI is Professor in the Department of Computer Science at The University of Western Ontario, London, Ontario, Canada. She received her M. Sc. in 1987 from the Univesity of Bucharest, Romania, and her Ph. D. in 1991 from the University of Turku, Finland. Her current research focuses on theoretical aspects of bioinformation and biocomputation, including models of cellular computation, nanocomputation by DNA self-assembly, and Watson-Crick complementarity in formal languages.

SHINNOSUKE SEKI received his Ph. D. in 2010 from the University of Western Ontario under Lila Kari. His research interests include algorithmic self-assembly and algorithmic drug design. He also has a deep interest in history, especially ancient Roman history. He is now working at the Department of Information and Computer Science, Aalto University.