Algorithmica (2013) 66:1–17
DOI 10.1007/s00453-012-9620-z

Tile Complexity of Approximate Squares

Harish Chandran · Nikhil Gopalkrishnan · John Reif

Received: 3 February 2011 / Accepted: 8 February 2012 / Published online: 22 February 2012

Abstract The standard Tile Assembly Model (TAM) of Winfree (Algorithmic self-assembly of DNA, Ph.D. thesis, 1998) is a mathematical theory of crystal aggregations via monomer additions with applications to the emerging science of DNA self-assembly. Self-assembly under the rules of this model is programmable and can perform Turing universal computation. Many variations of this model have been proposed and the canonical problem of assembling squares has been studied extensively.

We consider the problem of building approximate squares in TAM. Given any $\varepsilon \in (0, \frac{1}{4}]$ we show how to construct squares whose sides are within $(1 \pm \varepsilon)N$ of any given positive integer N using $O(\frac{\log \frac{1}{\varepsilon}}{\log\log \frac{1}{\varepsilon}} + \frac{\log\log \varepsilon N}{\log\log\log \varepsilon N})$ tile types. We prove a matching lower bound by showing that $\Omega(\frac{\log \frac{1}{\varepsilon}}{\log\log \frac{1}{\varepsilon}} + \frac{\log\log \varepsilon N}{\log\log\log \varepsilon N})$ tile types are necessary almost always to build squares of required approximate dimensions. In comparison, the optimal construction for a square of side exactly N in TAM uses $O(\frac{\log N}{\log\log N})$ tile types.

The question of constructing approximate squares has been recently studied in a modified tile assembly model involving concentration programming. All our results are trivially translated into the concentration programming model by assuming arbitrary (non-zero) concentrations for our tile types. Indeed, the non-zero concentrations could be chosen by an adversary and our results would still hold.

H. Chandran · N. Gopalkrishnan (✉) · J. Reif
Department of Computer Science, Duke University, Durham, NC 27708, USA
e-mail: nikhil@cs.duke.edu

H. Chandran
e-mail: harish@cs.duke.edu

J. Reif
e-mail: reif@cs.duke.edu

J. Reif
Adjunct, Computing and Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia

Our construction can get highly accurate squares using very few tile types and are feasible starting from values of N that are orders of magnitude smaller than the best comparable constructions previously suggested. At an accuracy of $\varepsilon = 0.01$, the number of tile types used to achieve a square of size 10^7 is just 58 and our constructions are proven to work for all $N \geq 13130$. If the concentrations of the tile types are carefully chosen, we prove that our construction assembles an $L \times L$ square in optimal assembly time $O(L)$ where $(1-\varepsilon)N \leq L \leq (1+\varepsilon)N$.

Keywords Self-assembly · Tile assembly model · Program size complexity · Approximation algorithms

1 Introduction

Self-assembly is a fundamental pervasive natural phenomenon that gives rise to complex structures and functions. It describes processes in which a disordered system of pre-existing components form organized structures as a consequence of specific, local interactions among the components themselves, without any external direction. In its most complex form, self-assembly encompasses the processes involved in growth and reproduction of higher order life. A simpler example of self-assembly is the orderly growth of crystals. In the laboratory, self-assembly techniques have produced increasingly complex structures (see [3, 11, 15, 22, 25] for some illustrative examples) and dynamical systems (see [12, 36–38]). While a surprising degree of control is exhibited in self-assembled systems in nature, a certain degree of flexibility is inherent in many systems. This suggests that self-assembly processes need not exactly match the desired outcome and approximate notions may be sufficient. The roots of attempts to model and study self-assembly begin with the study of tilings. In this paper we explore the notion of approximation for tilings of a square. The problem of constructing approximate squares in Tile Assembly Models (TAM) has received a lot of attention recently [5, 13, 18], but most of this work has been in alternate models of TAM. We show in this work that the original TAM of [31] allows practical constructions for approximate squares. We also prove the optimality of our constructions in this model.

1.1 Tiling Systems that Model Self-assembly

A *Wang tile* [29] is an oriented unit square with a pad associated with each side. Any two tiles with the same pads on all corresponding sides are said to be of the same *tile type*. Given a finite set S of Wang tiles types, a valid arrangement of S on a planar unit square grid consists of copies of Wang tiles from the set S such that abutting pads of all pairs of neighboring tiles match. The *tiling* or *domino* problem for a set of Wang tiles is: can tiles from S (chosen with replacement) be arranged to cover the entire planar grid? Berger [7] proved the undecidability of the tiling problem by reducing the halting problem to it. Robinson [24] gave an alternative proof involving a simulation of any single tape deterministic Turing Machine by some Wang tiling system. Garey and Johnson [16] and Papadimitriou and Lewis [20] proved that the problem of tiling a finite rectangle is **NP**-complete. These results paved the way for Wang tiling systems to be used for computation. But, Wang tilings do not model coordinated growth and hence do not describe complex self-assembly processes. Winfree [34] extended Wang tilings to the Tile Assembly Model (TAM) with a view to model self-assembly processes, laying a theoretical foundation for a form of DNA based computation (see [27] for details of the model), in particular, molecular computation via assembly of DNA lattices with tiles in the form of DNA motifs.

Rothemund and Winfree [27] define *tile complexity* of a shape as the minimum number of tile types for assembling that shape. Tile complexity, apart from capturing the information complexity of shapes, is also important as there exist fundamental limits on the number of tile types one can design using DNA sequences of fixed length. Various ingenious constructions for shapes like squares (see [1]), rectangles and computations like counting (see [4]) etc. exist in this model.

1.2 Outline of Results

A canonical problem in TAM and its extensions is the assembly of squares from unit sized square tiles. This simple but fundamental question in self-assembly has helped formulate notions of tile complexity and running time. Becker et al. [5] and later Kao and Schweller [18] used this question to introduce notions of probabilistic assembly and approximate shapes. Kao and Schweller [18] achieve, with arbitrarily high probability, squares of required approximate dimensions using $O(1)$ tile types in an extension of TAM where tile attachments are probabilistic based on their relative concentrations. Doty [13] extended this result to achieve, with arbitrarily high probability, squares of required exact dimensions using $O(1)$ tile types in the same model. Only a constant number of tile types are required because all the information content of the square is encoded in the tile concentration assignment. Thus, the number of tile types is not a measure of the information complexity of the shape.

We wish to separate the notion of approximate shapes from probabilistic assembly and consider them independently. Chandran et al. [8] introduced notions of probabilistic assembly that encode concentrations implicitly in tile set specification. Section 5 deals with purely deterministic assembly of approximate squares in the standard TAM where the minimum number of tile types for forming an approximate square is closely related to the descriptional complexity of the shape. Given any $\varepsilon \in (0, \frac{1}{4}]$ we show in Sect. 5.1 how to construct squares whose sides are within $(1 \pm \varepsilon)N$ of any given positive integer N using $O(\frac{\log \frac{1}{\varepsilon}}{\log\log \frac{1}{\varepsilon}} + \frac{\log\log \varepsilon N}{\log\log\log \varepsilon N})$ tile types. We prove a matching lower bound in Sect. 5.3 by showing that $\Omega(\frac{\log \frac{1}{\varepsilon}}{\log\log \frac{1}{\varepsilon}} + \frac{\log\log \varepsilon N}{\log\log\log \varepsilon N})$ tile types are necessary almost always to build squares of required approximate dimensions. In comparison, the optimal construction for a square of side exactly N uses $O(\frac{\log N}{\log\log N})$ tile types (see [1]). If the concentrations of the tile types are carefully chosen, we prove that our construction assembles an $L \times L$ square in optimal assembly time $O(L)$ where $(1-\varepsilon)N \leq L \leq (1+\varepsilon)N$.

2 Related Work in Constructing Approximate Squares

Various techniques have been suggested previously to build approximate squares in tile assembly models. In contrast to our work, all these involve working in modified domains of the abstract Tile Assembly Model (TAM) of [30]. In the *temperature programming* modification, squares of arbitrary size are achieved (see [17]) using a constant number of tile types by affecting a sequence of temperature changes. In the *staged assembly* modification (see [10]), the assembly process is performed in a sequence of stages allowing construction of arbitrary shapes using a constant number of tile types. In contrast to our work, in both these models the computational complexity of the target shape is encoded at least partly within the sequence of operations performed on sets of tiles as opposed to encoding all complexity purely in some attribute of the tile set itself.

In the *concentration programming* modification, the assembly is not deterministic as more than one tile type can compete for binding at the same location in an assembly and the relative probabilities of attachment are governed by tile type concentrations in the solution. Programming the relative concentrations of tile types controls the size and shape of assemblies created. Among the drawbacks of this model (in addition to the usual drawbacks of TAM): depletion of concentration of tiles as they attach to the growing assembly are usually not modeled and arbitrarily precise concentrations are often required. In the full version of his paper Doty [13] discusses these problems and suggests methods to get around them. Becker et al. [5] suggested using concentration programming to build squares of expected size N for all N using just 5 tile types. However, each square size is geometrically distributed about its expectation N and hence has high variance. Kao and Schweller [18] through an ingenious use of concentration programming gave tile sets that for any precision ε, certainty δ and sufficiently large integer N achieve, with probability at least $1-\delta$, squares of size L, where $(1-\varepsilon)N \leq L \leq (1+\varepsilon)N$, using a constant number of tile types. However, for a $(0.01, 0.01)$ approximation their constructions are only proven to work, according to their analysis, for squares of size $N \geq 10^{13}$. Even for tiles as small as 1 nanometer across, $N \geq 10^{13}$ works out to 10 kilometer size squares. Doty [13] significantly improved this result, achieving exact squares, with arbitrarily high probability $1-\delta$, of size N for sufficiently large N. For $\delta = 0.01$ certainty his constructions are proven to work, according to his analysis, for squares of size $N \geq 10^7$, which for tiles of size 1 nanometer works out to 1 centimeter. In simulations, he suggests that lower values of N may suffice. The number of tile types used in his construction for a $\delta = 0.01$ certainty is ≈ 5000.

A direct comparison of our results to these results via concentration programming is not formally correct as the models of self-assembly are different. However, all our results are trivially translated into the concentration programming model by assuming arbitrary concentrations for our tile types. Indeed, the concentrations could be chosen by an adversary and our results would still hold, as long as the concentrations chosen are non-zero. We can thus compare our results to those obtained via concentration programming. For an accuracy $\varepsilon = 0.01$, our constructions are proven to work for all $N \geq 13130$. For tiles of size 1 nanometer this works out to about 13 micrometers. The number of tile types used to achieve a square of size 10^7 at an accuracy of $\varepsilon = 0.01$

is just 58. If we increase our accuracy to $\varepsilon = 0.001$, we still need only 66 tile types to construct squares of size 10^7. Thus, we can get highly accurate squares using very few tile types and our constructions, even when translated into the concentration programming model, are feasible starting from values of N that are orders of magnitude smaller than the best comparable constructions previously demonstrated.

3 Tile Assembly Model

This section describes the abstract Tile Assembly Model (TAM). For a complete description of the model see [27]. Readers familiar with the TAM may skip this section. Consider the two-dimensional grid of integers $\mathbb{Z} \times \mathbb{Z}$ on which our tiles lie. The directions $\mathfrak{D} = \{\text{North}, \text{South}, \text{East}, \text{West}\}$ are functions from $\mathbb{Z} \times \mathbb{Z}$ to $\mathbb{Z} \times \mathbb{Z}$, with $\text{North}(x, y) = (x, y + 1)$, $\text{South}(x, y) = (x, y - 1)$, $\text{East}(x, y) = (x + 1, y)$ and $\text{West}(x, y) = (x - 1, y)$. We say that (x, y) and (x', y') are neighbors if $(x', y') \in \{\text{North}(x, y), \text{South}(x, y), \text{East}(x, y), \text{West}(x, y)\}$. Note that $\text{North}^{-1} = \text{South}$, $\text{East}^{-1} = \text{West}$ and vice versa. $\mathbb{N}$ is the set of natural numbers.

A *Wang tile* over the finite set of distinct *pads* Σ is a unit square whose four sides have pads (not necessarily distinct) from the set Σ. Formally, a tile t is an ordered pair of pads $(N_t, E_t, S_t, W_t) \in \Sigma^4$ indicating pad types on the North, East, South and West sides respectively. Note that tiles are not necessarily isomorphic under in place rotation or reflection. For each tile t, we define $\text{pad}_{\text{North}}(t) = N_t$, $\text{pad}_{\text{South}}(t) = S_t$, $\text{pad}_{\text{East}}(t) = E_t$ and $\text{pad}_{\text{West}}(t) = W_t$. Σ contains a special *null pad*, denoted by ϕ. The *empty* tile (ϕ, ϕ, ϕ, ϕ) represents the absence of any tile. Pads determine when two tiles attach. A function $g : \Sigma \times \Sigma \to \mathbb{N}$ is a *pad strength function* if it satisfies $\forall x, y \in \Sigma, g(x, y) = g(y, x)$ and $g(\phi, x) = 0$.

A *tiling system*, $\mathbb{T}$, is a tuple $\langle T, S, g, \tau \rangle$ where T containing the empty tile is the finite set of tile types, $S \subset T$ is the set of *seed* tiles, g is the pad strength function and τ is a global thermodynamic parameter. A *configuration* of T is a function $A : \mathbb{Z} \times \mathbb{Z} \to T$ with $A(0, 0) = s$ for some $s \in S$. For $D \in \mathfrak{D}$ we say the tiles at (x, y) and $D(x, y)$ *interact* with binding strength $g(\text{pad}_D(A(x, y)), \text{pad}_{D^{-1}}(A(D(x, y))))$. For all $s \in S$ a *start* configuration start_s is given by $\text{start}_s(0, 0) = s$ and $\text{start}_s(x, y) = \text{empty}$ otherwise. A tile $t \in T$ is said to τ-stably attach to a configuration A at position (x, y) iff $A(x, y) = \textit{empty}$ and $\sum_{D \in \mathfrak{D}} g(\text{pad}_D(t), \text{pad}_{D^{-1}}(A(D(x, y)))) \geq \tau$. *Self-assembly* is defined by a relation between configurations, $A \to B$, if there exists a tile $t \in T$ that τ-stably attaches to A to form B. We define $A \xrightarrow{*} B$ as the transitive closure of $\to$ and say B is *derived* from A. A configuration B is *produced* if $\text{start}_s \xrightarrow{*} B$ for some $s \in S$. A configuration is *terminal* if it is produced from start_s for some $s \in S$ and no configuration can be derived from it. $\text{Term}(\mathbb{T})$ is the set of terminal configurations of $\mathbb{T}$. In TAM, a terminal configuration is thought of as the output of a tiling system given a seed tile $s \in S$. We restrict ourselves to tile systems with a unique seed tile. Also, our pad strength function is restricted to be *diagonal*, i.e., $\forall x \neq y : g(x, y) = 0$. Finally, we ask that there be a unique terminal configuration, $\text{Term}(\mathbb{T}) = \{C\}$. Note that different attachment orders are allowed as long as they produce the same terminal configuration.

DNA nanostructures can physically realize TAM as shown in [33] with the DX tile and in [19] with the TX tile. Like the square tile in TAM, the DX and TX have *pads* that specify their interaction with other tiles. The pads are DNA sequences that attach via hybridization of complimentary nucleotides. Reference [21] is an experimental demonstration of computation via tile assembly using TX tiles. In [35] parallel XOR computation is executed in the test-tube using Winfree's DX tile. Other simple computations have also been demonstrated. However, large and more complex computations are beset by errors and error correction remains a challenge towards general computing using DNA tiles.

4 Constructing Squares in the Tile Assembly Model

A terminal configuration C is called an $N \times N$ square iff there exists a position (x_0, y_0) such that $\forall x, y \in \mathbb{Z}$ satisfying $x_0 \leq x < x_0 + N$ and $y_0 \leq y < y_0 + N$: $C(x, y) \neq \mathit{empty}$. For all other positions, $C(x, y) = \mathit{empty}$. Tile set complexity for self-assembly of squares at $\tau = 1$ is relatively uninteresting, requiring $\Omega(N)$ tile types (see [14, 27]) and hence we consider only $\tau \geq 2$ systems in this work.

Rothemund and Winfree [27] demonstrated a construction for forming $N \times N$ squares for any given N using only $O(\log N)$ tile types. The tiles self-assemble into a fixed width binary counter in the shape of a long rectangle using $\lceil \log N \rceil$ tile types. A fixed set of distinct *filler* tiles are then used to complete the rectangle into a square of the desired size. They further proved, using information theoretic arguments, that any construction for forming $N \times N$ squares needs $\Omega(\frac{\log N}{\log \log N})$ tile types for almost all N. This lower bound was achieved by [1] who demonstrated a construction using $O(\frac{\log N}{\log \log N})$ tile types for any given N using a more efficient counter. These results show that the program size complexity of self-assembled squares are closely related to the minimum tile set used to construct them.

Various modifications and additions to the tile assembly model have been proposed (see [2, 5, 10]). The minimum tile set required to construct squares in these models have been studied, with dramatic improvements in certain cases. However, the relationship between number of tile types and the information content of the square assembly program is not preserved in most of these models. We restrict ourselves to the standard tile assembly model of [27] in this work.

4.1 Self-assembly of a Binary Counter

Figure 1 illustrates a self-assembled infinite binary counter at $\tau = 2$, first described in [27]. The system consists of seven tiles. L, S (the seed tile) and R are the frame tiles which delimit the counter. L and R each bind to S and with themselves with strength 2 pads, indicated by a dark grey strip on their sides. They do not appear anywhere else in the counter. Two tiles with face label[1] 0 and two with face label 1 perform binary counting. Thus the face label of the tile at the $(i + 1)$th row (from the bottom) and $(j + 1)$th column (from the right) is the jth least significant bit of the counter after i steps of counting, assuming counting initiated at zero. They have

[1]The face labels merely depict the bits of the counter and do not play any role in tile binding. Here, they are identical to the North pad type.

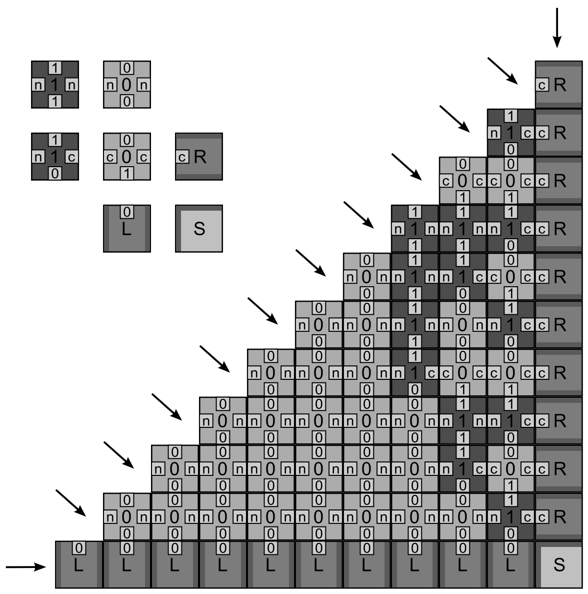
Fig. 1 Counter of Rothemund and Winfree using seven tile types

strength one pads on all four sides. One of the pad labels from {0, 1} occur on the North and South sides and indicate bit value. One of the pad labels from $\{c, n\}$ occur on the East and West sides and indicate presence (c) or absence (n) of carry. The addition of the 0 and 1 tiles to the growing assembly occurs via binding interactions on its East and South sides. Note that both these pads must match for binding to take place at $\tau = 2$. Each of the 0 and 1 tiles are designed such that the sum of the bits represented by the South and East pads is propagated on the North pad as a bit value (0 or 1) and on the East pad as the carry (n or c). A fixed width version of this binary counter is achieved with minor modifications as discussed in the next section.

4.2 Squares Using a Finite Counter

Figure 2(i) generically illustrates the construction for an $N \times N$ square using $O(\log N)$ tile types (from [27]). At the heart of the construction is a binary counter similar to the one described in Sect. 4.1. The chief difference in the two counters is that the new counter can start counting from any given n-bit binary number to 2^n and halts. This introduces additional complexity in the tile system necessitating additional tile types. The counter, a thin rectangle, is then extended into a square of the required size using diagonal building and filler tiles.

The system consists of a set of input tiles specific to the target dimension of the square and 23 other tiles independent of the dimension. Recall that the thermodynamic parameter τ is set to 2. The seed row tiles (input tiles), $\Theta(\log N)$ in number, account for almost all of the descriptional complexity of the shape. They initiate the assembly by encoding the binary number to start counting from. The leftmost (rightmost) column of the counter is built using border tiles with no binding attachments to other counter tiles on the West (East) side thus restricting the counter to a finite

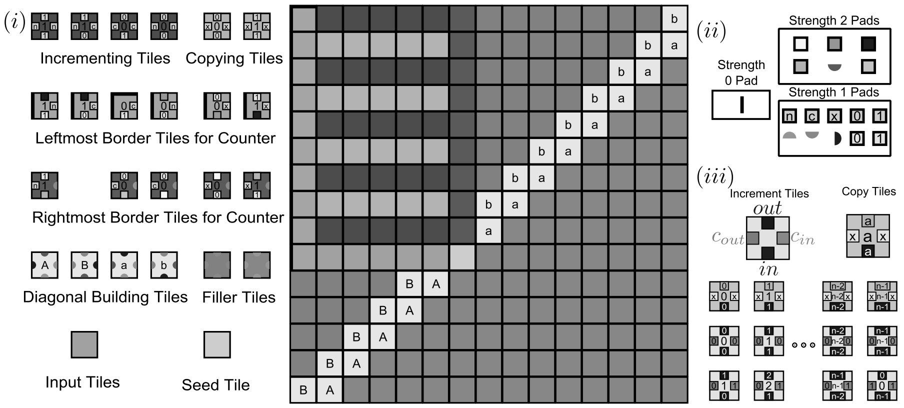

Fig. 2 (**i**) $N \times N$ square using $O(\log N)$ tile types. (**ii**) Pads for $N \times N$ square using $O(\log N)$ tile types. (**iii**) Increment and copy tiles for base d counter. The border tiles are not shown. The number of tile types is $\Theta(d)$

width. A pair of rows of the counter encode the same binary number, where the top row copies the bottom row. The copy row conserves the fixed width nature of the counter while at the same time propagating the appropriate carry bit. This copying is achieved by two copy tiles while the increment operation is implemented by four increment tiles analogous to those in Fig. 1. The counter halts when the leftmost bit rolls over from 1 to 0, indicated by a tile with a null pad (ϕ) on its North side. There are two pairs of diagonal building tiles which form a staggered diagonal from the seed tile. The rest of the square is completed with two filler tiles giving an $N \times N$ square for any given N using $O(\log N)$ tile types. For more details see [27].

The tile complexity for assembling an $N \times N$ square was reduced to $O(\frac{\log N}{\log\log N})$ in [1], asymptotically matching the information theoretic almost always lower bound of $\Omega(\frac{\log N}{\log\log N})$. Analogous to the construction in [27], a counter is used to assemble a thin rectangle that is extended to a square. The difference being, counting is now performed in a higher base which reduces the number of tile types required to form the seed row. Recall that the seed row needs to be composed of distinct tile types which accounts for most of the tile complexity of the system. Figure 2(iii) illustrates $\Theta(d)$ tile types for fixed width counting in base d, apart from the seed row. By choosing $d = \lceil \frac{\log N}{\log\log N} \rceil$ we can assemble $N \times N$ squares for any given N using $O(\frac{\log N}{\log\log N})$ tile types. For more details see [1]. We shall use this basic construction in the next section for building squares of dimension close to a given target N using exponentially lower number of tile types.

5 Constructing ε-Approximate Squares

Optimally concise tile sets for self-assembly of squares of exact size N have been well studied and the lower bound achieved, leaving no more room for improvement beyond constant factors. However, if we are allowed some error in side length, say by an additive fractional factor ε ($\pm\varepsilon N$), we can design significantly more concise tile sets for squares. We describe such tile sets in this section. Our tile sets satisfy exactly the rules of the Tile Assembly Model of [27]. In particular, each tile set is diagonal and produces a unique terminal configuration in the shape of a square. Thus, the improved tile complexity we achieve can be compared with the optimal tile set for exactly sized self-assembled squares and the difference attributed precisely to the notion of approximation introduced by us.

5.1 Construction

An $L \times L$ square is called an *ε-approximation* of an $N \times N$ square iff $(1-\varepsilon)N \leq L \leq (1+\varepsilon)N$. Since the size of any self-assembled square is an integer, the error term εN vanishes for $N < \frac{1}{\varepsilon}$ and hence we only consider $N \geq \frac{1}{\varepsilon}$. We now describe our construction for an ε-approximation of an $N \times N$ square using $O(\frac{\log \frac{1}{\varepsilon}}{\log\log \frac{1}{\varepsilon}} + \frac{\log\log \varepsilon N}{\log\log\log \varepsilon N})$ tile types.

We observe that an ε-approximation of a number N can be obtained by zeroing out an appropriate number of lower order bits in its binary representation. If we are presented with such a seed row, we can use the construction set out in [27] to obtain an ε-approximation of an $N \times N$ square. However, it is expensive to use distinct tile types to construct such a seed row as it is about $\Theta(\log N)$ tiles long. The only information content in the lower order 0 bits is *the number of such* 0*'s*. We therefore use a *minor counter* to obtain the correct number of 0's, and distinct tile types for the higher order bits. These put together act as a seed assembly for a *major counter* which is then completed to a square by staggered diagonal and filler tiles. Such a construction while close to being correct does not yield the proper approximation as we need to budget for the extra width due to the minor counter and the major counter. The construction is also not optimal as counting is done in binary, resulting in a longer seed row. We fix the latter problem by carefully choosing a higher base for the counter and the former by adjusting what number to start counting from. The technical details of the adjusted construction are set out below.

Let d be a carefully chosen integer that we shall use as a base for a counter assembly described later. Given any sufficiently large number N, we construct a square of size L with $(1-\varepsilon)N \leq L \leq (1+\varepsilon)N$. Let $N_1 = \lfloor (N - \lfloor \log_d N \rfloor)/2 \rfloor$. Consider the encoding of N_1 in base d (d-ary encoding): $b_{n-1}b_{n-2}\dots b_0$ where $n = \lfloor \log_d N_1 \rfloor + 1$. From N_1 we derive N_2 by setting all but the left most $k = \lceil \log_d \frac{1}{\varepsilon} \rceil + 1$ symbols in the d-ary encoding of N_1 to 0 to get $N_2 = b_{n-1}b_{n-2}\dots b_{n-k}0\dots 0$. We attempt to achieve a square of size $2N_2 + n$ (but will slightly overshoot). Let $N_3 = 1\underbrace{00\dots 0}_{n} - N_2 = c_{n-1}c_{n-2}\dots c_{n-k}\underbrace{00\dots 0}_{n-k}$.

Our construction has four logical components, described below. The final shape obtained is the union of these components. For logical clarity, the reader may think of the assembly proceeding component by component, though parts of some components may begin assembling before other components assemble completely.

1. *L-shaped seed assembly* (*Fig.* 3(ii)): Each tile in this component is unique and attaches to two adjacent tiles in the same component via strength 2 pads. The k tiles of the vertical arm of the L-shaped assembly sequentially encode the symbols

$c_{n-1}, c_{n-2}, \ldots, c_{n-k}$, with c_{n-1} being the Northern most. The tiles of the horizontal arm encode a *seed row* for a base $m = \lceil \frac{\log(n-k)}{\log\log(n-k)} \rceil$ counter (called the *minor counter*) with vertical length $n-k$. Thus, there are $\Theta(\frac{\log(n-k)}{\log\log(n-k)})$ tiles on the horizontal arm, which encode $e_{m-1}, e_{m-2}, \ldots, e_0$, East to West. The number of tile types used in this component is $\Theta(k + \frac{\log(n-k)}{\log\log(n-k)})$.

2. *Minor counter* (*Fig.* 3(iii)): This counter is seeded by the horizontal arm of the L-shaped assembly. It forms a rectangle of width m and height $n-k$ (excluding the *seed row*) by counting in base $\Theta\lceil \frac{\log(n-k)}{\log\log(n-k)} \rceil$. Note that the East pads of each tile that assembles on the Eastern border of the counter can be easily set to a unique special pad. We will use this pad to attach a unique tile encoding the symbol 0 (in base d) all along the Eastern border of the minor counter. The vertical arm of the L-shaped assembly and this vertical row of 0 tiles together represent the d-ary encoding of the number N_3. The number of tile types used in this component is $\Theta(\frac{\log(n-k)}{\log\log(n-k)})$.
3. *Major counter* (*Fig.* 3(v)): The major counter uses the n length vertical *seed column* representing the d-ary encoding of N_3 to count upto $1\underbrace{00\ldots0}_{n}$ in base d to get an $n \times 2N_2$ rectangle (inclusive of the *seed column*). Note that the Northeastern most tile of the counter occurs exactly once in the assembly, and its North and East pads are not involved in the logic of the counter assembly. The number of tile types used in this component is $\Theta(d)$, since we count in base d.
4. *Diagonal and filler tiles* (*Fig.* 4): Having constructed the major counter, we now complete the square. The bounded rectangular area to the North and West of the L-shaped seed assembly can be filled in using a single tile type whose North and South pads are identical to the North pad of the tiles forming the horizontal arm of the L-shaped seed assembly and West and East pads are identical to the West pad of the tiles forming the vertical arm of the L-shaped seed. Both the pads have strength one. From the northeastern most tile in the major counter, we will initiate an AB-type diagonal for the square (see Sect. 4.2) in two directions, Northwest and Southeast, and then use filler tiles to complete the square. We require pads on the Eastern boundary of the major counter that are distinct from any pads appearing within the major counter so that the filler tiles do not interfere with the major counter assembly. This is done by adding a single vertical column of tiles to the East of the Eastern boundary of the major counter. A unique distinct strength 2 East pad on the second from top tile in the vertical column initiates the diagonal in the Southeastern direction, while a unique distinct strength 2 North pad on the Northeastern most pad of the major counter initiates the diagonal in the Northeastern direction. The complete square has size $L = m + n + 2N_2$.

   All the East pads on the vertical column (except the top two) are identical and match the corresponding pads on the filler tiles. All the North pads on the Northern boundary of the major counter are identical (except the Eastern most) and match the corresponding pads on the filler tiles. The vertical column to the East of the major counter can be achieved with just three tile types, two for the top two tiles and one for the rest. Both the parts of the AB diagonals require two tile types each. Two types of filler tile types are required, one fills the area above the diagonal and

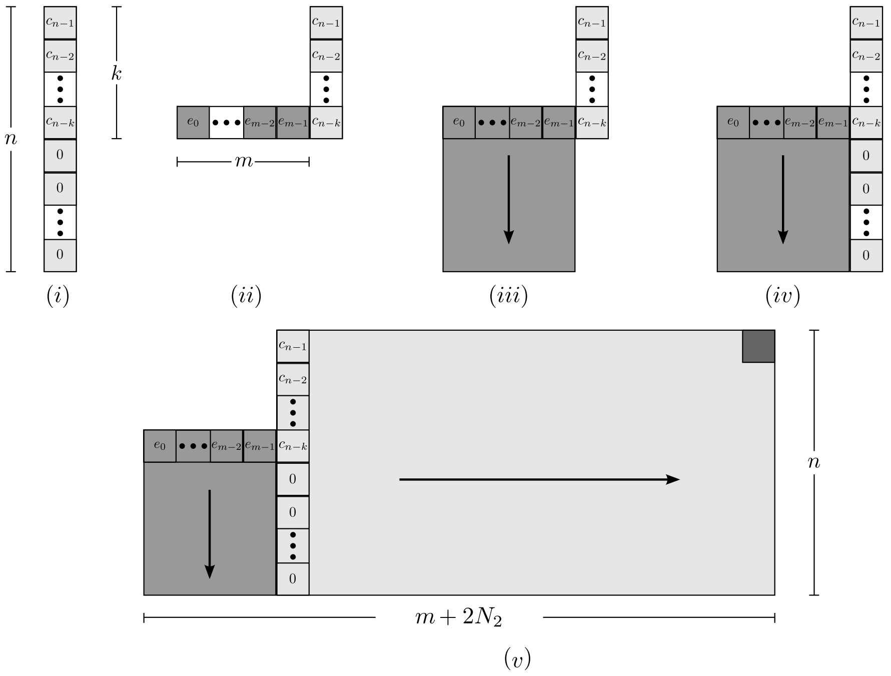

Fig. 3 Components of the construction: (**i**) *Seed column* for the major counter. (**ii**) L-shaped seed assembly. (**iii**) Assembly of the minor counter. (**iv**) Completing the *seed column* using the 0 tile type. (**v**) Assembly of the major counter

another fills the area below. Thus, the number of tile types used in this component is 10.

5.2 Analysis

We will first argue that our construction yields a unique terminal assembly in the shape of a square. Then we will show that an appropriate choice of the base d for the major counter gives us a construction of the claimed size and number of tile types. The first component, the L-shaped assembly, clearly uniquely assembles into the shape illustrated in Fig. 3. The minor counter and the major counter are implemented exactly as described in Fig. 2(iii), apart from some blunt pads on the boundary being changed to distinct pads that do not have any binding with any other pads involved in the assembly of the counter. The correctness of these counters have already been proved previously (see [1, 27]). By using distinct pad types for the two counters, and the L-shaped seed assembly, we ensure they do not have any undesired interactions. Finally, the diagonal and filler tiles are few in number (only 10) and their correctness can be verified by the reader.

Lemma 1 *The construction described in Sect. 5.1 produces a unique terminal $L \times L$ square.*

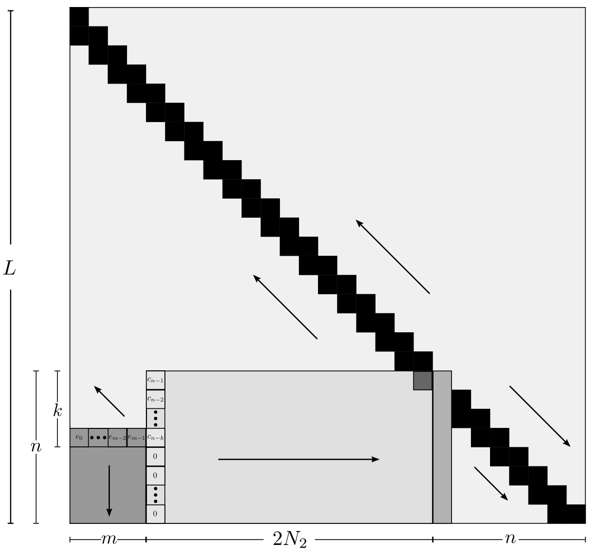

Fig. 4 Diagonal and filler tiles complete the approximate square of length $L = 2N_2 + m + n$

The choice of the base d affects the number of tile types in the vertical arm of the L-shaped seed assembly and also the number of tile types used in the major counter. A bigger value for d reduces the number of tile types used for the vertical arm, but increases the number of tile types in the major counter, while a smaller value does the opposite. We balance these two factors by choosing $d = \lceil \frac{\log \frac{1}{\varepsilon}}{\log\log \frac{1}{\varepsilon}} \rceil$. Note that this value, and hence the number of tile types for the major counter, is independent of N.

Theorem 1 *For the construction described above, for all $\varepsilon \in (0, \frac{1}{4}]$ and sufficiently large N, a choice of $d = \lceil \frac{\log \frac{1}{\varepsilon}}{\log\log \frac{1}{\varepsilon}} \rceil$ produces a unique terminal $L \times L$ square where $(1-\varepsilon)N \le L \le (1+\varepsilon)N$ and uses $O(\frac{\log \frac{1}{\varepsilon}}{\log\log \frac{1}{\varepsilon}} + \frac{\log\log \varepsilon N}{\log\log\log \varepsilon N})$ tile types.*

Proof Recall that $L = m + 2N_2 + n$; $d = \lceil \frac{\log \frac{1}{\varepsilon}}{\log\log \frac{1}{\varepsilon}} \rceil$; $N_1 = \lfloor \frac{N - \lfloor \log_d N \rfloor}{2} \rfloor$; $m = \lceil \frac{\log(n-k)}{\log\log(n-k)} \rceil$; $n = \lfloor \log_d N_1 \rfloor + 1$ and $k = \lceil \log_d \frac{1}{\varepsilon} \rceil + 1$. Also, $n - k = \lfloor \log_d N_1 \rfloor + 1 - \lceil \log_d \frac{1}{\varepsilon} \rceil - 1 \le \log_d(\varepsilon N_1)$.

Claim 1: For all $\varepsilon \in (0, \frac{1}{4}]$ and all sufficiently large N, $L \le (1+\varepsilon)N$.

We first upper bound n, m and $2N_2$:

1. $n = \lfloor \log_d N_1 \rfloor + 1 \le \log_d(N - \lfloor \log_d N \rfloor) - \log_d 2 + 1 \le \log_d(N - \lfloor \log_d N \rfloor) + 1 \le \log_d N + 1$.
2. $m = \lceil \frac{\log(n-k)}{\log\log(n-k)} \rceil \le 1 + \log(n-k) \le \log(\log_d(\varepsilon N_1) + 1) + 1 \le 2\log\log_d(\varepsilon N_1) \le 2\log\log_d(\varepsilon N)$.
3. $2N_2 \le 2N_1 \le N - \lfloor \log_d N \rfloor \le N - \log_d N + 1$.

So $L = m + 2N_2 + n \le 2\log\log_d(\varepsilon N) + N - \log_d N + 1 + \log_d N + 1 = N + 2\log\log_d(\varepsilon N) + 2 \le (1+\varepsilon)N$ for sufficiently large N.

Claim 2: For all $\varepsilon \in (0, \frac{1}{4}]$ and all sufficiently large N, $L \geq (1-\varepsilon)N$.

Recall that $N_1 = b_{n-1}b_{n-2}\ldots b_0$; $N_2 = b_{n-1}b_{n-2}\ldots b_{n-k}\underbrace{0\ldots0}_{n-k}$ and so $N_1 - N_2 = b_{n-k-1}b_{n-k-2}\ldots b_0 \leq d^{n-k} - 1 \leq \varepsilon N_1 - 1$. Hence $2N_2 \geq 2N_1(1-\varepsilon) + 2$. Also, $n = \lfloor \log_d N_1 \rfloor + 1 \geq \log_d N_1 \geq \log_d(\frac{N - \lfloor \log_d N \rfloor - 2}{2}) = \log_d(N - \lfloor \log_d N \rfloor - 2) - \log_d 2 \geq \log_d \frac{N}{2} - \log_d 2 \geq \log_d N - 2 \geq \lfloor \log_d N \rfloor - 2$ and $2N_1 = 2\lfloor \frac{N - \lfloor \log_d N \rfloor}{2} \rfloor \geq N - \lfloor \log_d N \rfloor - 2$. Thus $L = m + 2N_2 + n \geq 2 + 2N_2 + n \geq 2 + 2N_1(1-\varepsilon) + 2 + \lfloor \log_d N_1 \rfloor + 1 \geq 2 + 2N_1(1-\varepsilon) + 2 + \lfloor \log_d N \rfloor - 2 \geq 2 + 2N_1 - 2\varepsilon N_1 + \lfloor \log_d N \rfloor \geq N - 2\varepsilon N_1 \geq N - \varepsilon N = (1-\varepsilon)N$ for sufficiently large N.

Claim 3: The number of tile types used in the construction is $O(\frac{\log \frac{1}{\varepsilon}}{\log\log \frac{1}{\varepsilon}} + \frac{\log\log \varepsilon N}{\log\log\log \varepsilon N})$.

The number of tile types used in the L-shaped seed assembly is $\Theta(k + \frac{\log(n-k)}{\log\log(n-k)})$, the number of tile types used in the minor counter is $\Theta(\frac{\log(n-k)}{\log\log(n-k)})$ and in the major counter is $\Theta(d)$. The diagonal and filler assemblies use 10 tile types. Thus the total number of tile types for our construction is $O(\frac{\log \frac{1}{\varepsilon}}{\log\log \frac{1}{\varepsilon}} + \frac{\log\log \varepsilon N}{\log\log\log \varepsilon N})$. □

5.3 Lower Bound

In this section we derive a lower bound on the number of tile types required to assemble an ε-approximation of an $N \times N$ square. We use information theoretic arguments and hence our bounds hold for almost all $N \in \mathbb{N}$. We are interested in approximation ratios less than $\frac{1}{4}$. In the following lemma we relate the lower bound function obtained in Theorem 1 from information theoretic arguments to the upper bound from Theorem 1.

Lemma 2 *For all* $\varepsilon \in (0, \frac{1}{4}]$: $\frac{\log(\frac{1}{\varepsilon} + \log N)}{\log(\log \frac{1}{\varepsilon} + \log\log \varepsilon N)} = \Omega(\frac{\log \frac{1}{\varepsilon}}{\log\log \frac{1}{\varepsilon}} + \frac{\log\log \varepsilon N}{\log\log\log \varepsilon N})$.

Proof We first choose a sufficiently large N for all the subsequent inequalities in the proof to hold. We then prove the claim separately for $\frac{1}{\varepsilon} \leq \log N$ and $\frac{1}{\varepsilon} \geq \log N$. In the inequalities that follow, we repeatedly use substitutions that rely on facts like log is a non-negative, increasing function, $\varepsilon < 1$ and $a^2 \geq 2a$ for $a \geq 2$. We also use the substitutions $\frac{1}{\varepsilon} \leq \log N$ (first case) and $\frac{1}{\varepsilon} \geq \log N$ (second case). For $\frac{1}{\varepsilon} \leq \log N$, we have $\frac{\log \frac{1}{\varepsilon}}{\log\log \frac{1}{\varepsilon}} + \frac{\log\log \varepsilon N}{\log\log\log \varepsilon N} \leq \frac{\log \frac{1}{\varepsilon}}{\log\log \frac{1}{\varepsilon}} + \frac{\log\log N}{\log\log\log N} \leq \frac{2\log\log N}{\log\log\log N} \leq \frac{2\log(\frac{1}{\varepsilon} + \log N)}{\log\log\log N} = \frac{4\log(\frac{1}{\varepsilon} + \log N)}{2\log\log\log N} \leq \frac{4\log(\frac{1}{\varepsilon} + \log N)}{\log(\log\log N + \log \frac{1}{\varepsilon})} \leq 4\frac{\log(\frac{1}{\varepsilon} + \log N)}{\log(\log \frac{1}{\varepsilon} + \log\log \varepsilon N)}$. Similarly for $\frac{1}{\varepsilon} \geq \log N$, we have $\frac{\log \frac{1}{\varepsilon}}{\log\log \frac{1}{\varepsilon}} + \frac{\log\log \varepsilon N}{\log\log\log \varepsilon N} \leq \frac{\log \frac{1}{\varepsilon}}{\log\log \frac{1}{\varepsilon}} + \frac{\log\log N}{\log\log\log N} \leq \frac{2\log \frac{1}{\varepsilon}}{\log\log \frac{1}{\varepsilon}} \leq \frac{2\log(\frac{1}{\varepsilon} + \log N)}{\log\log \frac{1}{\varepsilon}} = \frac{4\log(\frac{1}{\varepsilon} + \log N)}{2\log\log \frac{1}{\varepsilon}} \leq \frac{4\log(\frac{1}{\varepsilon} + \log N)}{\log(\log\log N + \log \frac{1}{\varepsilon})} \leq 4\frac{\log(\frac{1}{\varepsilon} + \log N)}{\log(\log \frac{1}{\varepsilon} + \log\log \varepsilon N)}$. □

Theorem 2 *For all* $\varepsilon \in (0, \frac{1}{4})$, *the number of tile types required to assemble an* ε*-approximation of an* $N \times N$ *square is* $\Omega(\frac{\log \frac{1}{\varepsilon}}{\log\log \frac{1}{\varepsilon}} + \frac{\log\log \varepsilon N}{\log\log\log \varepsilon N})$ *for almost all* N.

Proof Let L be the size of an ε-approximation of an $N \times N$ square. Then we know that $(1-\varepsilon)N \le L \le (1+\varepsilon)N$ which implies $\frac{N}{2} \le L \le 2N$ since $\varepsilon < \frac{1}{2}$. So the number of bits in the binary encoding of L, $n = \lfloor \log L \rfloor + 1 \ge \lfloor \log N \rfloor$. Any construction that builds an ε-approximation of an $N \times N$ square is an implicit encoding of the number n. Also, from L, we can recover the first $\lceil \log \frac{1}{\varepsilon} \rceil$ bits of N. Let these bits represent the number $e \ge 2^{\lceil \log \frac{1}{\varepsilon} \rceil - 1} \ge \frac{1}{2\varepsilon}$. Thus, from the assembled square, we implicitly get the number $n' = n + e \ge \lfloor \log N \rfloor + \frac{1}{2\varepsilon}$. Thus, by arguments similar to [27], we have $(1-\delta)\log n' < \mathcal{K}_U(n') \le |p_{SA}| + f(K_\varepsilon(N))$ for any $0 < \delta < 1$ and almost all N. Here, $\mathcal{K}_U(n')$ is the Kolmogorov complexity of the binary number n' with respect to the universal Turing machine U, $K_\varepsilon(N)$ is the minimum number of tile types to form an ε-approximation of an $N \times N$ square, p_{SA} is a program that on input a tile set for an $N \times N$ ε-approximate square outputs n' and $f(K_\varepsilon(N)) = O(K_\varepsilon(N)\log(K_\varepsilon(N)))$ is the size of the encoding of the tile set of size $K_\varepsilon(N)$. Thus, $(1-\delta)\log(\lfloor \log N \rfloor + \frac{1}{2\varepsilon}) \le c_1 + c_2 K_\varepsilon(N)\log(K_\varepsilon(N))$. From Theorem 1, $K_\varepsilon(N) \le O(\frac{\log \frac{1}{\varepsilon}}{\log\log \frac{1}{\varepsilon}} + \frac{\log\log \varepsilon N}{\log\log\log \varepsilon N})$ and so $(1-\delta)\log(\lfloor \log N \rfloor + \frac{1}{2\varepsilon}) \le c_1 + c_2 K_\varepsilon(N)\log(\frac{\log \frac{1}{\varepsilon}}{\log\log \frac{1}{\varepsilon}} + \frac{\log\log \varepsilon N}{\log\log\log \varepsilon N})$. Note that $\lfloor \log N \rfloor + \frac{1}{2\varepsilon} \ge \log N + \frac{1}{\varepsilon}$ and $\frac{\log \frac{1}{\varepsilon}}{\log\log \frac{1}{\varepsilon}} + \frac{\log\log \varepsilon N}{\log\log\log \varepsilon N} \le \log \frac{1}{\varepsilon} + \log\log \varepsilon N$ for all sufficiently large N. Using these two inequalities and transposing all terms except $K_\varepsilon(N)$ to one side, we get $K_\varepsilon(N) = \Omega(\frac{\log(\frac{1}{\varepsilon} + \log N)}{\log(\log \frac{1}{\varepsilon} + \log\log \varepsilon N)})$ almost always and hence by Lemma 2, $K_\varepsilon(N) = \Omega(\frac{\log \frac{1}{\varepsilon}}{\log\log \frac{1}{\varepsilon}} + \frac{\log\log \varepsilon N}{\log\log\log \varepsilon N})$ almost always. □

6 Optimal Assembly Times of ε-Approximation of $N \times N$ Squares

Adleman et al. [1] introduced the notion of assembly time of a tile system. Informally, their model envisages a partial assembly floating in solution and encountering various tiles. If an encountered tile can stick to the assembly, it does so and the rates (and probabilities) of such additions are governed by the relative concentration of tiles. Note that they assume an inexhaustible supply of each tile type and thus the relative concentrations are unchanged through the assembly process.

Formally, the assembly process can be modeled as a continuous time Markov process where the configurations reachable from the seed are in one-to-one correspondence with the states of the Markov process, with the seed as the initial state and the terminal assembly as the unique sink state. An edge exists between two states A and B if the addition of a tile type t to the configuration corresponding to A produces the configuration corresponding to B, with a transition rate equal to the relative concentration of t. The time to reach the sink state from the initial state is a random variable. The expectation of this random variable is defined to be the assembly time of the tile system. Given a fixed temperature, the time complexity of constructing a given shape is the smallest assembly time among all tile systems that produce the shape at that temperature. Adleman et al. [1] proved that the time complexity of assembling an $N \times N$ square at any temperature in TAM is $\Omega(N)$ and gave a construction for squares that achieved asymptotically optimal running time and program size complexity at $\tau = 3$. Subsequently [6] also achieved a time optimal construction for squares but at the lower temperature of $\tau = 2$ and a faster (by a constant factor) assembly time.

The assembly time of our construction can be improved without affecting the asymptotic tile complexity by using fast parallel binary counters instead of our current higher base counter. However, encoding the seed for the counter in binary increases the number of tile types. Instead, we can retain the encoding of the seed row in a higher base and convert it to binary using a simple base converter. Both the fast parallel binary counter and base converter are described in [1]. Our construction for an ε-approximation of an $N \times N$ square will contain seven components: an L-shaped seed assembly, two base converters, two fast parallel binary counters, diagonal component and filler component. By allocating a combined relative concentration of $\frac{1}{7}$ to each component, we can ensure that each assembles in time $O(L)$ to give an overall assembly time of $O(L)$, where $(1-\varepsilon)N \leq L \leq (1+\varepsilon)N$. The details of the tile set for the fast parallel counter and base converter can be found in [1].

Most of the tiles that make up each approximately-sized square are either filler tiles, diagonal tiles or counter tiles. It is useful for these sub-assemblies to be constructed from a fixed set of tiles that do not depend on ε and N. There are 10 fixed diagonal and filler tile types and we can use 10 fixed tile types for each of the two parallel binary counters and hence the major portion of any approximate square can be constructed from this fixed set of 30 tile types which can be synthesized in bulk.

7 Discussion and Open Problems

Laboratory implementation of simple constructions from the Tile Assembly Model using DNA have been successful carried out [26]. However these implementations have been beset by errors due to spurious nucleation and coordinated binding errors. Various error correction and prevention techniques have been studied to deal with these challenges (see [9, 23, 28, 32]). Our construction suffers from these same kind of errors and the same error correcting techniques proposed previously apply to our constructions. For any given ε, our constructions have exponentially smaller tile sets than the optimal tile sets for exact squares for all large enough N. Such large constructions are not currently achievable in the laboratory, though the field is progressing rapidly. Our notion of approximation led us to construct target shapes that were approximately-sized but in the shape of squares. Under different notions of approximation, one could analyze similar questions. Another challenge is to develop notions of approximation for arbitrary shapes, constructing these and lower bounding their tile complexity in TAM.

Acknowledgements We thank our reviewers for pointing out errors and for comments that improve the readability of the paper. This work was supported by NSF EMT Grants CCF-0829797 and CCF-0829798.

References

1. Adleman, L., Cheng, Q., Goel, A., Huang, M.D.: Running time and program size for self-assembled squares. In: Symposium on Theory of Computing, pp. 740–748 (2001)
2. Aggarwal, G., Cheng, Q., Goldwasser, M.H., Kao, M.Y., de Espanes, P.M., Schweller, R.T.: Complexities for generalized models of self-assembly. SIAM J. Comput. **34**(6), 1493–1515 (2005)
3. Andersen, E., Dong, M., Nielsen, M., Jahn, K., Subramani, R., Mamdouh, W., Golas, M., Sander, B., Stark, H., Oliveira, C., Pedersen, J.S., Birkedal, V., Besenbacher, F., Gothelf, K., Kjems, J.: Self-assembly of a nanoscale DNA box with a controllable lid. Nature **459**(7243), 73–76 (2009)
4. Barish, R., Rothemund, P., Winfree, E.: Two computational primitives for algorithmic self-assembly: copying and counting. Nano Lett. **5**, 2586–2592 (2005)
5. Becker, F., Rapaport, I., Remila, E.: Self-assemblying classes of shapes with a minimum number of tiles, and in optimal time. In: Foundations of Software Technology and Theoretical Computer Science, pp. 45–56 (2006)
6. Becker, F., Remila, E., Schabanel, N.: Time optimal self-assembling of 2D and 3D shapes: the case of squares and cubes. In: Goel, A., Simmel, F., Sosík, P. (eds.) DNA Computing. Lecture Notes in Computer Science, vol. 5347, pp. 144–155. Springer, Berlin (2009)
7. Berger, R.: The undecidability of the domino problem. Mem. Am. Math. Soc. **66**, 1–72 (1966)
8. Chandran, H., Gopalkrishnan, N., Reif, J.: The tile complexity of linear assemblies. In: International Colloquium on Automata, Languages and Programming, pp. 235–253 (2009)
9. Chen, H.L., Goel, A.: Error free self-assembly using error prone tiles. In: Ferretti, C., Mauri, G., Zandron, C. (eds.) DNA Computing. Lecture Notes in Computer Science, vol. 3384, pp. 702–707. Springer, Berlin (2005)
10. Demaine, E., Demaine, M., Fekete, S., Ishaque, M., Rafalin, E., Schweller, R., Souvaine, D.: Staged self-assembly: nanomanufacture of arbitrary shapes with $O(1)$ glues. Nat. Comput. **7**(3), 347–370 (2008)
11. Dietz, H., Douglas, S., Shih, W.: Folding DNA into twisted and curved nanoscale shapes. Science **325**(5941), 725–730 (2009)
12. Dirks, R., Pierce, N.: Triggered amplification by hybridization chain reaction. Proc. Natl. Acad. Sci. USA **101**(43), 15275–15278 (2004)
13. Doty, D.: Randomized self-assembly for exact shapes. SIAM J. Comput. **39**(8), 3521–3552 (2010)
14. Doty, D., Patitz, M., Summers, S.: Limitations of self-assembly at temperature 1. Theor. Comput. Sci. **412**(1–2), 145–158 (2011)
15. Douglas, S., Dietz, H., Liedl, T., Hogberg, B., Graf, F., Shih, W.: Self-assembly of DNA into nanoscale three-dimensional shapes. Nature **459**(7245), 414–418 (2009)
16. Garey, M., Johnson, D.: Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, New York (1981)
17. Kao, M.Y., Schweller, R.: Reducing tile complexity for self-assembly through temperature programming. In: Symposium on Discrete Algorithms, pp. 571–580 (2006)
18. Kao, M.Y., Schweller, R.: Randomized self-assembly for approximate shapes. In: International Colloquium on Automata, Languages and Programming, pp. 370–384 (2008)
19. LaBean, T., Yan, H., Kopatsch, J., Liu, F., Winfree, E., Reif, J., Seeman, N.: Construction, analysis, ligation, and self-assembly of DNA triple crossover complexes. J. Am. Chem. Soc. **122**(9), 1848–1860 (2000)
20. Lewis, H., Papadimitriou, C.: Elements of the Theory of Computation. Prentice Hall, New York (1981)
21. Mao, C., Labean, T., Reif, J., Seeman, N.: Logical computation using algorithmic self-assembly of DNA triple-crossover molecules. Nature **407**, 493–496 (2000)
22. Park, S.H., Yin, P., Liu, Y., Reif, J., LaBean, T., Yan, H.: Programmable DNA self-assemblies for nanoscale organization of ligands and proteins. Nano Lett. **5**, 729–733 (2005)
23. Reif, J., Sahu, S., Yin, P.: Compact error-resilient computational DNA tiling assemblies. In: Ferretti, C., Mauri, G., Zandron, C. (eds.) DNA Computing. Lecture Notes in Computer Science, vol. 3384, pp. 293–307. Springer, Berlin (2005)
24. Robinson, R.: Undecidability and nonperiodicity for tilings of the plane. Invent. Math. **12**, 177–209 (1971)
25. Rothemund, P.: Folding DNA to create nanoscale shapes and patterns. Nature **440**, 297–302 (2006)
26. Rothemund, P., Papadakis, N., Winfree, E.: Algorithmic self-assembly of DNA Sierpinski triangles. PLoS Biology **2**(12), e424 (2004), pp. 2041–2053

27. Rothemund, P., Winfree, E.: The program-size complexity of self-assembled squares. In: Symposium on Theory of Computing, pp. 459–468 (2000)
28. Schulman, R., Winfree, E.: Programmable control of nucleation for algorithmic self-assembly. SIAM J. Comput. **39**(4), 1581–1616 (2009)
29. Wang, H.: Proving theorems by pattern recognition II. Bell Syst. Tech. J. **40**, 1–41 (1961)
30. Winfree, E.: On the computational power of DNA annealing and ligation. In: DNA Based Computers. DIMACS, vol. 27, pp. 199–221. Am. Math. Soc., Providence (1995)
31. Winfree, E.: Algorithmic self-assembly of DNA. Ph.D. thesis, California Institute of Technology (1998)
32. Winfree, E., Bekbolatov, R.: Proofreading tile sets: error correction for algorithmic self-assembly. In: Chen, J., Reif, J. (eds.) DNA Computing, Lecture Notes in Computer Science, vol. 2943, pp. 1980–1981. Springer, Berlin (2004)
33. Winfree, E., Liu, F., Wenzler, L., Seeman, N.: Design and self-assembly of two-dimensional DNA crystals. Nature **394**, 539–544 (1998)
34. Winfree, E., Yang, X., Seeman, N.: Universal computation via self-assembly of DNA: some theory and experiments. In: DNA Based Computers II. DIMACS, vol. 44, pp. 191–213. Am. Math. Soc., Providence (1996)
35. Yan, H., Feng, L., LaBean, T., Reif, J.: Parallel molecular computation of pair-wise XOR using DNA string tile. J. Am. Chem. Soc. **125**(47), 14246–14247 (2003)
36. Yin, P., Choi, H., Calvert, C., Pierce, N.: Programming biomolecular self-assembly pathways. Nature **451**(7176), 318–322 (2008)
37. Yin, P., Yan, H., Daniell, X., Turberfield, A., Reif, J.: A unidirectional DNA Walker moving autonomously along a linear track. Angew. Chem., Int. Ed. **116**(37), 5014–5019 (2004)
38. Zhang, D., Turberfield, A., Yurke, B., Winfree, E.: Engineering entropy-driven reactions and networks catalyzed by DNA. Science **318**, 1121–1125 (2007)