Step-wise tile assembly with a constant number of tile types

Ján Maňuch[1,2], Ladislav Stacho[1], and Christine Stoll[1]

[1] Department of Mathematics, Simon Fraser University, Canada
[2] Department of Computer Science, University of British Columbia, Canada

Abstract. In this paper, we study the step-wise tile assembly model introduced by (Reif, 1999) in which shape is assembled in multiple steps. In each step the partially built structure is exposed to a new set of tiles. We show that an $N \times N$ square can be assembled in N steps using a constant number of tile types. In general, we show that a constant number of tile types (24) is sufficient to assemble a large class of shapes in n steps, where n is the number of tiles of the shape. This class includes all shapes obtained from any shape by scaling by a factor of 2, in which case only 14 tile types suffices. For general shapes, we show that the tile complexity of this model is related to the monotone connected node search number of a spanning tree of the shape assuming the number is steps is n.

1 Introduction

Self-assembly is the process by which simple parts autonomously assemble into larger, more complex objects. Self-assembly occurs in nature, for example, when atoms combine to form molecules, and molecules combine to form crystals. It has been suggested that intricate self-assembly schemes will ultimately be useful for circuit fabrication, nano-robotics, DNA computing, and amorphous computing (Abelson et al., 2000; Gomez-Lopez et al., 1996; Reif, 1999; Winfree et al., 1996).

The standard model to study the process of self-assembly is the Tile Assembly Model proposed by (Rothemund and Winfree, 2000) which considers the assembly of square blocks called "tiles" and a set of glues called "binding domains". Each of the four sides of a tile can have a glue on it that determines interactions with neighboring tiles. It is assumed that there is an infinite supply of tiles of each tile type. The process of self-assembly is initiated by a single seed tile and proceeds by attaching tiles one by one. A tile can only bind to the growing complex if it binds strongly enough, as determined by the *temperature* τ.

Branched DNA molecules (Seeman, 1998) provide a direct physical motivation for this model. DNA double-crossover molecules, each bearing four "sticky ends" analogous to the four sides of a tile, have been designed to self-assemble into a periodic two dimensional lattice (LaBean et al., 2000; Mao et al., 2000; Rothemund et al., 2004; Winfree et al., 1998). The binding interactions between double-crossover molecules may be redesigned by changing the base sequence of their sticky ends, thus allowing arbitrary sets of tiles to be investigated in the laboratory. (Demaine et al., 2008) mentions a possibility of implementing Wang tiles using protein-based designs, where unit-length nanorods (made of proteins) are joined at right angles at their midpoints to form a plus sign. Protein nanorod structures are, unlike DNA based assemblies, very rigid. It is believed that this rigidity will allow them to be used for the nanomanufacture of macroscale objects.

Reif (1999) informally introduced a modification to the standard model of self-assembly, called *step-wise tile assembly model*, which, instead of having a single set of tiles, uses several sets of tiles applied sequentially. During the assembly process, the seed tile is immersed into one of the sets, then after a sufficient time for the completion of this step's assembly, the unattached tiles are "washed away" leaving the partially constructed assembled shape which is then placed to the next set of tiles where it now acts as a seed and assembly continues. He used this model to implement local parallel biomolecular computations with the increased likelihood of success (compared to such implementations in the standard tile assembly model) for some fundamental problems that form the basis of many parallel computations.

In this paper we formally defined this model and study its implications to tile self-assemblies. In particular, we give an upper bound on the number of tile types required to uniquely assemble an arbitrary shape, if we do not require a bond between every two adjacent tiles is given ("partial connectivity"). We note that this bound is related to the monotone connected node search number (cf. (Kirousis and Papadimitriou, 1985)) of the underlying spanning tree. Lastly, we exhibit a large class of shapes that have constant tile complexity (24 tile types suffice). This class includes all shapes scaled by a factor of 2. We also show that for these scaled shapes (by a factor of 2), 14 tile types suffice. We note that the number of steps in these assemblies is linear in the size of the assembled shape.

A similar model, called the *staged assembly model*, has been proposed in (Demaine et al., 2008). The key differences between the step-wise tile assembly model and this model are that in step-wise tile assembly model growth occurs by sequential addition of single tiles to the seed configuration similarly as in the standard tile assembly model (Rothemund and Winfree, 2000), while in the stage assembly model two assemblies are allowed to attach to form a larger assembly. Moreover, the step-wise tile assembly model uses only one bin, while the staged assembly model uses multiples bins at each stage for later mixing of assembled supertiles (assemblies). In (Demaine et al., 2008), a bound of at most 52[⋆] tile types for uniquely assembling an arbitrary shape with partial connectivity. If scaling by a factor of two is allowed, then the authors show that any arbitrary hole-free shape can be assembled using at most 8 binding domains (and hence a constant number of tile types), and this construction ensures that there is a bond between every two adjacent tiles ("full connectivity").

[⋆] After personal communication with the authors of (Demaine et al., 2008) we were advised that original bound of 16 in the paper should be 52.

The comparison of our results with related previous results is shown in Table 1.

	Glues	Tiles	Bins	Stages/Steps	Scale	Conn.
$n \times n$ square						
(Rothemund and Winfree, 2000)	$\Theta(n^2)$	n^2	1	1	1	Full
(Demaine et al., 2008)	9	$O(1)$	$O(1)$	$O(\log n)$	1	Full
Theorem 1	6	9	1	$\Theta(n)$	1	Full
General shapes with n tiles						
(Demaine et al., 2008)	2	52	$O(\log n)$	$O(\text{diameter})$	1	Partial
Hole-free shapes ((Demaine et al., 2008))	8	$O(1)$	$O(n)$	$O(n)$	2	Full
Theorem 6	2	14	1	n	2	Partial
Shapes with rectangle decomposition (Theorem 5)	5	24	1	n	1	Partial

Table 1. Comparison of parameters in related stage assemblies at temperature 1.

The reader may be also interested in other models of self-assembly, for instance, flexible glue model, unique shape model, multiple temperature model, multiple tile model (Aggarwal et al., 2005). An extended abstract of this paper appeared in (Maňuch et al., 2009).

2 Description of the Step-Wise Tile Assembly Model

To define the standard and the step-wise tile assembly models we will follow the terminology of (Rothemund and Winfree, 2000). We will consider the square lattice, i.e., the graph with vertex set $\mathbb{Z}\times\mathbb{Z}$ and edge set $\{(u,v) : |u,v| = 1\}$. The directions $\mathcal{D} = \{N, E, S, W\}$ are used to indicate the natural directions in the lattice. Formally, they are functions from $\mathbb{Z}\times\mathbb{Z}$ to $\mathbb{Z}\times\mathbb{Z}$: $N(x,y) = (x, y+1)$, $E(x,y) = (x+1, y)$, $S(x,y) = (x, y-1)$, and $W(x,y) = (x-1, y)$. Note that $E^{-1} = W$ and $N^{-1} = S$.

A tile is a square with the north, east, south, and west edges labeled from some alphabet Σ of binding domains (glues). Formally, a tile t is a square represented by a 4-tuple $(t_N, t_E, t_S, t_W) \in \Sigma^4$ indicating the binding domains on the north, east, south, and west side, respectively. We will use *null* to indicate the lack of a binding domain, and will assume $\mathit{null} \in \Sigma$. The special tile $\mathit{empty} = (\mathit{null}, \mathit{null}, \mathit{null}, \mathit{null})$ represents an empty space when placed onto the grid. A *configuration* on a set of tiles T is a map $C : \mathbb{Z}\times\mathbb{Z} \to T$. We define the *vertex set* of configuration C as $V(C) = \{(x,y) : C(x,y) \neq \mathit{empty}\}$. A configuration C is *finite* if $V(C)$ is finite. We will refer to $C(x,y)$ as the tile at the vertex (x,y) in C. Given a configuration C and a set of vertices $V \subseteq \mathbb{Z}\times\mathbb{Z}$, a *sub-configuration* of C induced by V is the map $C[V] : \mathbb{Z}\times\mathbb{Z} \to T$ such that

$C[V](x,y) = C(x,y)$ for all $(x,y) \in V$, and $C[V](x,y) = \mathit{empty}$, otherwise. If G is any subgraph of the lattice graph, then we sometimes abuse the notation of $C[V(G)]$ to simply $C[G]$. Given two configurations C and D, we define their union to be the following map from $\mathbb{Z} \times \mathbb{Z}$ to $T \cup \{\infty\}$:

$$(C \cup D)(x,y) = \begin{cases} C(x,y) & \text{if } D(x,y) = \mathit{empty} \text{ or } C(x,y) = D(x,y), \\ D(x,y) & \text{if } C(x,y) = \mathit{empty} \text{ or } C(x,y) = D(x,y), \\ \infty & \text{otherwise.} \end{cases}$$

Note that $C \cup D$ is a configuration whenever it is a map to T. Equivalently, $C \cup D$ is not a configuration if there exists $(x,y) \in V(C) \cap V(D)$ such that $C(x,y) \neq D(x,y)$.

A *strength function* $g : \Sigma \times \Sigma \to \mathbb{N} = \{0,1,2,...\}$ measures the interaction strength between binding domains. We denote by s_Σ the strength function satisfying $s_\Sigma(\sigma,\sigma') = 1$, if $\sigma = \sigma' \neq \mathit{null}$, and $s_\Sigma(\sigma,\sigma') = 0$ otherwise. We call s_Σ a *simple* strength function. In this paper we restrict ourselves to simple strength functions.

Given a tile t, a configuration C, and a direction d, we denote the interaction strength in configuration C between tile t at position (x,y) and its respective neighboring tile by

$$g_d^C(t,x,y) = g(t_d, C(d(x,y))_{d^{-1}}).$$

Note that we do not require that $C(x,y) = t$. In particular, if $C(x,y) \neq t$, then $g_d^C(t,x,y)$, $d \in \mathcal{D}$ tells us how t would bind if it were in C at position (x,y). Given $(x,y) \in \mathbb{Z} \times \mathbb{Z}$ and $d \in \mathcal{D}$, we say that there is a *bond* between positions (x,y) and $d(x,y)$ in C if $g_d^C(C(x,y),x,y) > 0$. For a simple strength function g, this happens only if the binding domains on the abutting sides of the two tiles are the same. An *exposed side* of C is a pair $e = [(x,y),d]$, where $(x,y) \in V(C)$ and $d \in \mathcal{D}$ such that $C(x,y)_d \neq \mathit{null}$ and $C(d(x,y)) = \mathit{empty}$. Binding domain $C(x,y)_d$ is called the binding of exposed side e. If C has an exposed side at position (x,y) and direction d then C could make a bond on this side with a suitable tile.

Under the Tile Assembly Model a *tile assembly system* is a 5-tuple $\mathbf{T} = (\Sigma, T, \phi, g, \tau)$, where T is a finite set of tiles with binding domains from Σ and contains the tile *empty*, ϕ is a set of configurations on T called *seed configurations*, g is a strength function, and τ is a threshold parameter called *temperature*. We will be working with seed configurations consisting of a single tile; formally a configuration C_t, where $t \in T$, satisfying $C_t(0,0) = t$, and $C_t(x,y) = \mathit{empty}$ for all $(x,y) \in \mathbb{Z} \times \mathbb{Z} \backslash \{(0,0)\}$. To distinguish these tile assembly systems from step-wise tile assembly systems which we will define later, we will sometimes refer to the above tile assembly systems as *simple* tile assembly systems.

Self-assembly is now defined as a relation between configurations on T. Let C and D be two configurations of T, such that $C = D$ except at position (x,y), where $C(x,y) = \mathit{empty}$, and $D(x,y) = t$, for some $t \in T \backslash \{\mathit{empty}\}$. Then we write $C \rightarrow_{\mathbf{T}} D$, if

$$\sum_{d \in \mathcal{D}} g_d^C(t, x, y) \geq \tau.$$

This means that a tile can be added to a configuration at position (x, y), if and only if the sum of the interaction strengths of t at position (x, y) with its neighbors reaches or exceeds τ. For simple strength functions and $\tau = 1$, which are considered in this paper, $C \rightarrow_{\mathbf{T}} D$ if and only if there exists $d \in \mathcal{D}$ such that $g_d^C(t, x, y) = 1$. The relation $\rightarrow_{\mathbf{T}}^*$ is the reflexive transitive closure of $\rightarrow_{\mathbf{T}}$.

Of particular interest is a subclass of configurations that arises from the self-assembly process. A tile assembly system $\mathbf{T}$ and the relation $\rightarrow_{\mathbf{T}}^*$ define the partially ordered set of configurations called *assemblies* of $\mathbf{T}$: $Asmb(\mathbf{T}) = \{A : S \rightarrow_{\mathbf{T}}^* A$, where $S \in \phi\}$, and the set of *terminal assemblies* of $\mathbf{T}$: $Term(\mathbf{T}) = \{A \in Asmb(\mathbf{T}) : \nexists B$ such that $A \rightarrow_{\mathbf{T}}^+ B\}$. We say that a tile assembly system $\mathbf{T}$ *uniquely produces* A if $Term(\mathbf{T}) = \{A\}$.

We now extend the definition of a tile assembly system to allow for several sets of tiles sequentially applied on the growing configuration. A *step-wise tile assembly system* is a 5-tuple $\mathbf{T}_{step} = (\Sigma, \{T_i\}_{i=1}^k, \{C_t\}, g, \{\tau_i\}_{i=1}^k)$, where $\{T_i\}_{i=1}^k$ is a sequence of finite sets of tiles (each set including the tile *empty*), C_t is the initial seed configuration (consisting of the single tile t), g is a strength function, and $\{\tau_i\}_{i=1}^k$ is a sequence of temperatures. We define the (simple) tile assembly system at step 1 as $\mathbf{T}_1 = (\Sigma, T_1, \{C_t\}, g, \tau_1)$ and the (simple) tile assembly system at step i as $\mathbf{T}_i = (\Sigma, T_i, Term(\mathbf{T}_{i-1}), g, \tau_i)$ for $2 \leq i \leq k$. This allows us to view a step-wise tile assembly system as a sequence of simple tile assembly systems.

We define the set of terminal assemblies of $\mathbf{T}_{step}$ as $Term(\mathbf{T}_{step}) = Term(\mathbf{T}_k)$. Given a configuration A, we say that the step-wise tile assembly system $\mathbf{T}_{step}$ *uniquely produces* A, if $Term(\mathbf{T}_{step}) = \{A\}$. We are interested in the number of tile types used in a tile assembly system and define the *tile complexity* of $\mathbf{T}_{step}$ as $|\bigcup_{i=1}^k T_i| - 1$ (where we are subtracting 1 to exclude the *empty* tile).

3 Tile Complexity of Step-Wise Assembled Shapes

A *shape* S is a finite connected induced subgraph of the lattice. We say a configuration A *has shape* S, if $V(A) = V(S)$. We say that a tile assembly system uniquely produces shape S, if the the terminal assembly of the tile assembly system is unique and has shape S.

In the standard tile assembly model (where only a single set of tiles is allowed), N^2 distinct tile types are required to uniquely assemble an $N \times N$ full square at temperature 1 (Rothemund and Winfree, 2000), in contrast in the stage assembly model only a constant number of tile types. In the step-wise tile assembly model a constant number of tile types suffices as well.

Theorem 1. *A full $N \times N$ square can be uniquely assembled using 4 distinct non-empty tile types and $N - 1$ steps if N is odd, and 9 distinct non-empty tile types and $N + 1$ steps if N is even, under the step-wise tile assembly model with temperature 1.*

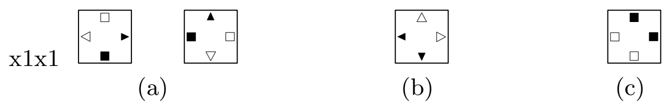


Fig. 1. The tile sets ((a) tile set S_1; (b) tile set S_2; and (c) tile set S_3) for assembling $N \times N$ squares, where N is odd.

Proof. Figure 1 shows the tile sets for assembling an $N \times N$ square, where N is odd. As we are working with temperature 1, all binding domains have strength 1. The tile from S_3 also serves as the seed tile. The tile set sequence $\{T_i\}_{i=1}^{N-1}$ is defined as follows:

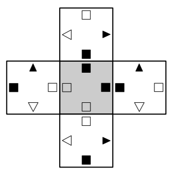
$$T_i = \begin{cases} S_1 & \text{if } i \text{ is odd}, \\ S_2 & \text{if } i \equiv 2 \pmod 4, \\ S_3 & \text{if } i \equiv 0 \pmod 4). \end{cases}$$

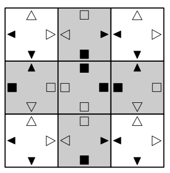
We further assume that each T_i contains the *empty* tile. Figure 2 shows the terminal assemblies for each step in the construction of a 5×5 square. Immersing the seed tile in the tile set S_1 causes the tiles of S_1 to attach to the northern, southern, eastern, and western boundary of the seed configuration, creating a "plus sign". Adding the tiles of S_2 to this terminal assembly causes the corners to fill in, creating a 3×3 square. In step $2k - 1$, $k \geq 2$, the tiles of S_1 attach to every second tile on the northern, southern, eastern, and western boundary of the seed square. Immersing this assembly now into the tile set T_{2k} causes the tile from T_{2k} to fill in every other tile on the boundary of the new seed assembly, creating a $(2k + 1) \times (2k + 1)$ square. Thus, T_{N-1} uniquely produces a $N \times N$ square.

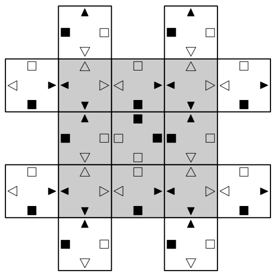
If we want to build an $N \times N$ square, where N is even, we first assemble an $(N - 1) \times (N - 1)$ square as above, and then use the tile sets E_1, E_2, and E_3 as illustrated in Figure 3, if N is divisible by four. Otherwise, if N is two modulo four, the binding domains exposed on the north and east side of the $(N-1) \times (N-1)$ square are ■ and ▲, and we make the following binding domain replacements to the tiles of E_1 and E_2: □ $\rightarrow$ ▲ and △ $\rightarrow$ ■. To construct the $N \times N$ square, we first place the $(N - 1) \times (N - 1)$ square into the tile set E_1. The resulting shape is immersed into the tile set E_2, and finally the resulting terminal assembly is placed into E_3. Figure 4 shows the assembly of a 4×4 square starting from the 3×3 square. □

We now give an upper bound on the tile complexity of arbitrary shapes in the step-wise tile assembly model at temperature 1. Before we can state the main result of this section, we need some more definitions.

A *caterpillar* K is a graph consisting of a single path, called the *spine*, and additional vertices attached to the spine by single edges, called *hairs*. We refer to these additional vertices as *hairtips*. Given a tree F, a caterpillar K with spine P and set of hairtips L is a *natural caterpillar with respect to F* if it is


(a) seed tile S_3 immersed in S_1


(b) terminal assembly from (a) immersed in S_2


(c) terminal assembly from (b) immersed in S_1

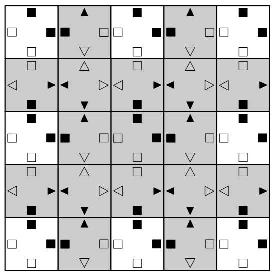

(d) terminal assembly from (c) immersed in S_3

Fig. 2. The terminal assemblies for each step in the construction of a 5×5 square. For each step, the tiles of seed of configuration are depicted gray and the added tiles in this step are depicted white. The seed configuration is the terminal assembly of the previous step, except in the first step, the seed configuration is the seed tile. To construct 7×7 square immerse the terminal assembly from (d) in S_1 and then thus produced terminal assembly in S_2.

a subgraph of F and all vertices in L are leaves in F. If F is clear from the context, we will omit reference to it. Moreover, K is *maximal* if it is natural and the endpoints of P are leaves of F. Given a caterpillar K with the spine P, we choose one of the endpoints of P and call it the *anchor* of K, or we say that K is *anchored* at v. Moreover, K is a *maximal caterpillar anchored at* v, if K is a natural caterpillar anchored at v and the other endpoint of the spine is a leaf of F. An illustration of these definitions is shown in Figure 5.

Given a tree F, the sequence of sets (called layers) of caterpillars $\{L_i\}_{i=1}^{\delta}$ forms a *maximal-layer decomposition* D_F of F if it satisfies all of the following:

(MLD1) All caterpillars in $\bigcup_{i=1}^{\delta} L_i$ are edge disjoint, and cover all edges of F.
(MLD2) Layer L_1 consists of a single maximal caterpillar of F.
(MLD3) For every $2 \leq i \leq \delta$, caterpillars in layer L_i are vertex disjoint.
(MLD4) For every $2 \leq i \leq \delta$, every caterpillar K in layer L_i is a maximal caterpillar anchored at a vertex v on the spine of some caterpillar in layer L_{i-1}.

An example of a maximal-layer decomposition is shown in Figure 6. Note that the anchor of a caterpillar K at layer L_i is either an internal vertex of the spine of a caterpillar K' or the anchor of K', where $K' \in L_{i-1}$. In the later case, the anchor of K is an internal vertex of the spine of a caterpillar in L_{i-2}.

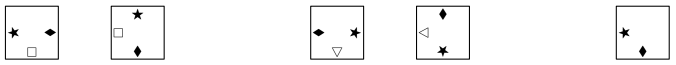

(a) (b) (c)

Fig. 3. Additional tile sets ((a) tile set E_1; (b) tile set E_2; and (c) tile set E_3) for assembling an $N \times N$ square, where N is divisible by four.

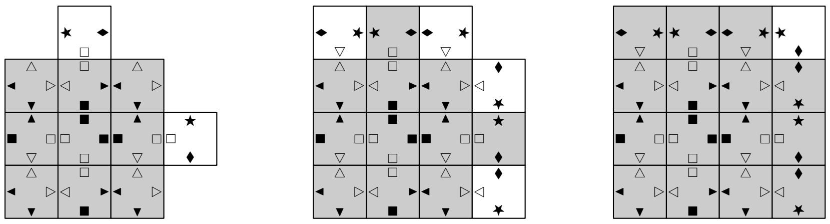

(a) terminal assembly from Fig. 2(b) immersed in E_1 (b) terminal assembly for (a) immersed in E_2 (c) temrinal assembly from (b) immersed in E_3

Fig. 4. The terminal assemblies of the last three steps in the construction of a 4×4 square.

A vertex v has *layer* i *in* D_F, if i is the smallest index such that v is a vertex of a caterpillar in L_i. The *depth of a maximal-layer decomposition* $D_F = \{L_i\}_{i=1}^{\delta}$ is δ. The *maximal-layer depth of a tree* F, denoted by MLD-depth(F), is the smallest δ such that F has a maximal-layer decomposition of depth δ. The *maximal-layer depth of a shape* S, MLD-depth(S), is the minimum maximal-layer depth over all spanning trees of S.

A vertex v is a *single anchor* if it is an anchor for exactly one caterpillar in D_F, and v is a *double anchor* if it is an anchor for exactly two caterpillars in D_F. Note that if F is a spanning tree of a shape S, then v cannot be an anchor for more than two caterpillars, since the degree $deg_F(v)$ of v in F is at most 4, and every anchor is an internal vertex of the spine of some caterpillar. Note that since for every $1 < i \leq \delta$, caterpillars of L_i are vertex disjoint, any vertex v of layer j that is a double anchor is an anchor for one caterpillar of L_{j+1} and one caterpillar of L_{j+2}; moreover, $\deg_F(v) = 4$ and the remaining two edges incident to v belong to a caterpillar of L_j. Note that in this case v cannot be a leaf of the caterpillar of L_j. We are ready to state and prove the main result of this section.

Theorem 2. *Any shape S that has a spanning tree F of maximal-layer depth δ can be uniquely produced by a step-wise tile assembly system at temperature 1 with tile complexity at most $68\delta + 8$ using $2\delta + 2$ non-null binding domains. Moreover, if the maximum degree of F is 3, S can be uniquely produced by a step-wise tile assembly system with tile complexity at most $44\delta + 8$ using $2\delta + 1$ non-null binding domains, and if the maximum degree of F is 2, only 14 tile*

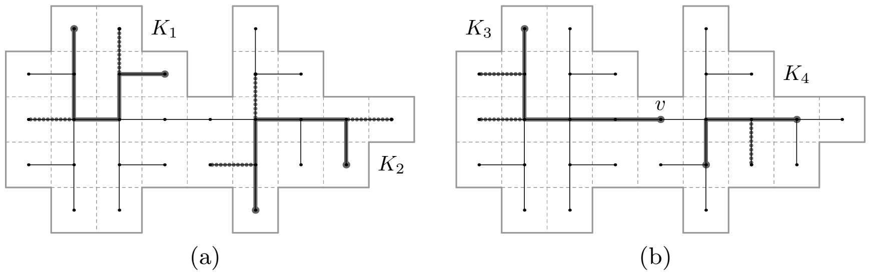


Fig. 5. Examples of natural caterpillar, maximal caterpillar and maximal caterpillar anchored at v. For each caterpillar, the spine is depicted as a solid line, the endpoints of the spine as circles and the hairs as dotted lines. Caterpillar K_1 is natural, while K_2 is not. Both K_1 and K_2 are maximal, while neither K_3 nor K_4 is. However, K_3 is a maximal caterpillar anchored at v.

types and 2 non-null binding domains suffice. In all cases, the number of steps is $|V(S)| - 1$.

Proof. Tiles of the shape S will be added in the order determined by a depth-first search of the vertices of F which starts at an endpoint of the spine of the caterpillar in the first layer and prefers to visit leaves and the caterpillars of higher layers. In each step, we will add exactly one tile by using a singleton tile set. The tile of layer i will use glues specific to this layer. Intuitively, along a spine of a caterpillar at layer L_i, we will alternate using glues a_i and b_i to connect tiles adjacent along the spine, and use glue a_{i+1} and a_{i+2} in any direction deviating from the spine which lead to a hair or a caterpillar of layer L_{i+1} or L_{i+2}. At any step, the exposed glues will be distinct.

Ordering of vertices of F. Let s_0 be one of the two endpoints of the spine of the caterpillar in L_1 (without loss of generality we assume that $s_0 = (0, 0)$). The sequence $\{s_j\}_{j=0}^{|V(S)|-1}$ is obtained by a depth-first search on the vertices of F starting at the vertex s_0. At each branching point the depth-first search will choose the next vertex to visit as follows: choose a non-visited neighbor that is a leaf, or, if none of the non-visited neighbors are leaves, choose a non-visited neighbor of the highest layer. Recall that the layer of a vertex is the smallest layer L_i which contains a caterpillar containing v. An example of the order of vertices determined by this procedure is depicted in Figure 7.

Binding domain types. Let $D_F = \{L_i\}_{i=1}^{\delta}$ be a maximal-layer decomposition of F of depth δ. In the step-wise tile assembly system we are constructing, each tile set will contain only a single non-empty tile type. The binding domains will be taken from the set $\Sigma = \{null, a_1, a_2, \ldots, a_{\delta+2}, b_1, b_2, \ldots, b_\delta\}$. We define a partial function $\overline{x}$ on Σ as follows: $\overline{a_i} = b_i$ and $\overline{b_i} = a_i$, for $1 \leq i \leq \delta$. We also define a *rank* on Σ as follows: $\text{rank}(a_i) = i$, $\text{rank}(b_i) = i$, and $\text{rank}(null) = 0$.

Tile sets. For every vertex s_j of F let $t^j = (t_N^j, t_E^j, t_S^j, t_W^j)$ be the tile that will be placed onto s_j. The tile set used in the step j of the construction will be the set $T_j = \{t^j, empty\}$. Next we specify the tile types t^j for each step

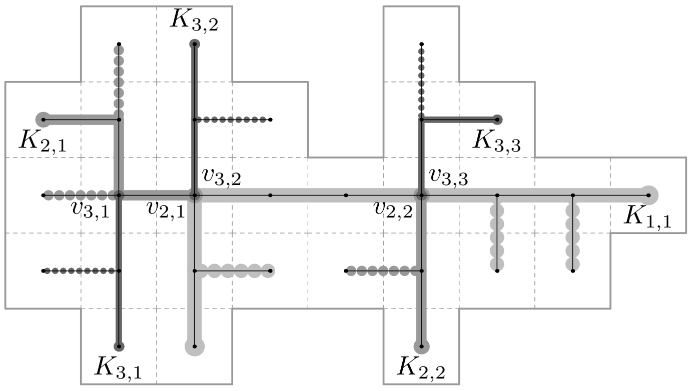


Fig. 6. An example of a maximal-layer decomposition of a tree spanning the given shape with layers $L_1 = \{K_{1,1}\}$, $L_2 = \{K_{2,1}, K_{2,2}\}$ and $L_3 = \{K_{3,1}, K_{3,2}, K_{3,3}\}$. The anchor of a caterpillar $K_{i,j}$ is denoted as $v_{i,j}$.

$j = 0, \ldots, |V(S)| - 1$ in this order. Note that s_1 is a neighbor of s_0. Let $d \in \mathcal{D}$ be the direction such that $s_1 = d(s_0)$. Set $t^0_d = a_1$ and $t^0_\alpha = null$, if $\alpha \neq d$.

For step $j > 1$, we will base the assignment of binding domains for every tile t^j on the order in which the neighbors of s_j appear in the order as follows. If s_l and s_m are two neighbors of s_j that both succeed s_j in the sequence obtained from the search algorithm, and s_l is visited before s_m, then the binding domain of tile t^j on the side adjacent to s_l has higher rank than the binding domain of t^j on the side adjacent to s_m. The binding domains of tile t^j will be determined by the properties of s_j in the maximal-layer decomposition D_F and the relative position of s_j with respect to its neighbors in the order. We distinguish fours cases on depending on degree of s_j.

Case $\deg_F(s_j) = 1$. Let s_k be the neighbor of s_j such that $k < j$. Let d be the direction such that $s_k = d(s_j)$. Set $t^j_d = t^k_{d^{-1}}$ (this ensures that the tile t^j can attach to the neighboring tile t^k), and $t^j_\alpha = null$, if $\alpha \neq d$.

Case $\deg_F(s_j) = 2$. Let s_k be the unique neighbor of s_j such that $k < j$. Note that s_{j+1} is also a neighbor of s_j, and, because $\deg_F(s_j) = 2$, s_k, s_j, and s_{j+1} all belong to the same caterpillar in some L_i. Let d_1, d_2 be directions such that $s_k = d_1(s_j)$, and $s_{j+1} = d_2(s_j)$. Set $t^j_{d_1} = t^k_{d_1^{-1}}$ (this ensures that the tile t^j can attach to its neighboring tile t^k), $t^j_{d_2} = \overline{t^j_{d_1}}$, and $t^j_\alpha = null$, if $\alpha \notin \{d_1, d_2\}$.

Case $\deg_T(s_j) = 3$. We distinguish two cases:

(i) s_j is not an anchor: Then s_j belongs to exactly one caterpillar K in layer L_i for some i. Two of its neighbors are on the spine of K and the third must be a leaf which is visited by the search algorithm immediately after s_j, and hence it is s_{j+1}. Let s_k and s_m be the other two neighbors of s_j such that $k < j$ and $m > j$ (see Figure 8(a)). Note that $s_m = s_{j+2}$ and that s_j and all its neighbors belong to the same caterpillar. Let d_1, d_2, d_3 be directions such that $d_1(s_j) = s_k$, $d_2(s_j) = s_{j+1}$, and $d_3(s_j) = s_m$. Set $t^j_{d_1} = t^k_{d_1^{-1}}$ (this

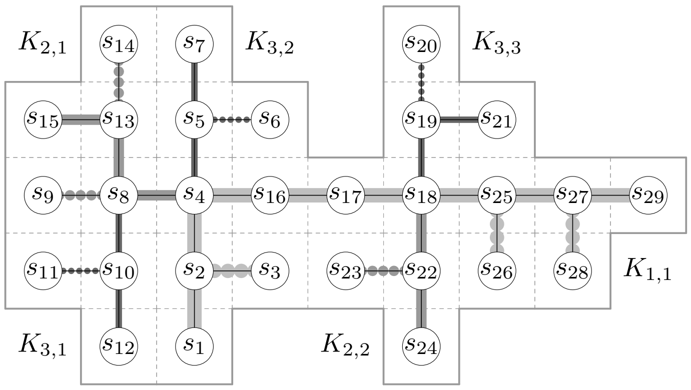


Fig. 7. The order of vertices determined by the depth first search algorithm based on the maximal-layer decomposition depicted in Figure 6.

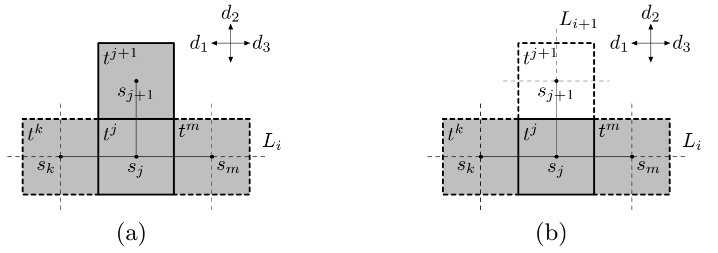


Fig. 8. Tiles for a vertex of degree 3. The tiles t^n are indicated together with the vertices s_n of the spanning tree of S. Dashed sides of a tile indicate that another tile could attach to that side. Dashed edges of the tree indicate how the tree might continue. The shading represents the caterpillar of layer i that the vertex s_j belongs to. The relative directions are indicated in the top right corner of each figure.

ensures that tile t^j can attach to the neighboring tile t^k), $t^j_{d_2} = a_{i+1}$ (hence using a new binding domain for hairs of K), $t^j_{d_3} = \overline{t^j_{d_1}}$, and $t^j_\alpha = null$, if $\alpha \notin \{d_1, d_2, d_3\}$.

(ii) s_j is an anchor: Then s_j must be a single anchor, since only vertices of degree four can be double anchors. Let s_j be an anchor for a caterpillar K' in L_{i+1}. It follows that s_j also belongs to a caterpillar K in L_i (see Figure 8(b)). Let s_k, s_m, d_1, d_2, and d_3 be defined as in case (i). Note that s_{j+1} is a neighbor of s_j and its layer is $i+1$. Set $t^j_{d_1} = t^k_{d_1^{-1}}$ (this ensures that tile t^j can attach to the neighboring tile t^k), $t^j_{d_2} = a_{i+1}$, (as the caterpillar K' will use a new pair of binding domains a_{i+1}, b_{i+1}), $t^j_{d_3} = \overline{t^j_{d_1}}$, and $t^j_\alpha = null$, if $\alpha \notin \{d_1, d_2, d_3\}$.

Case $\deg_F(s_j) = 4$. We distinguish three cases:

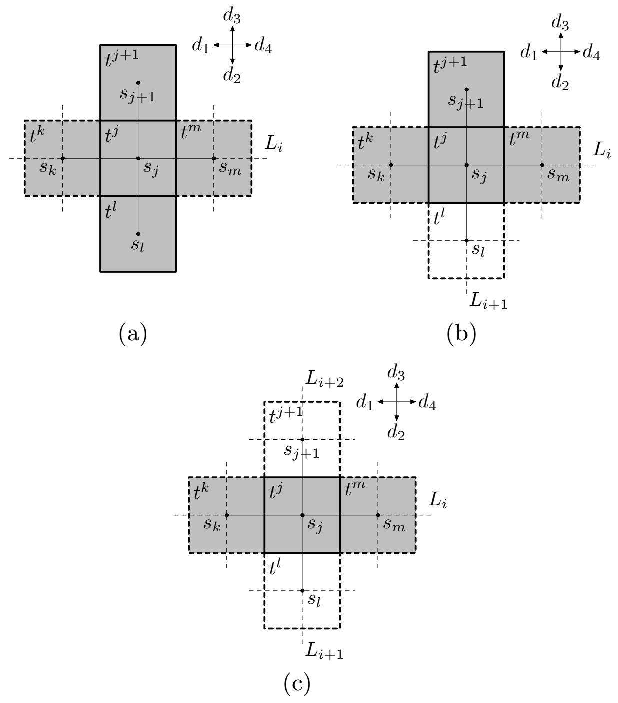


Fig. 9. Tiles for a vertex of degree 4. The tiles t^n are indicated together with the vertices s_n of the spanning tree of S. Dashed sides of a tile indicate that another tile could attach to that side. Dashed edges of the tree indicate how the tree might continue. The shading represents the caterpillar of layer i that the vertex s_j belongs to. The relative directions are indicated in the top right corner of each figure.

(i) s_j is not an anchor: Then two of the neighbors of s_j belong to the spine of some caterpillar K in L_i for some i, and the other two neighbors of s_j are leaves of K. Note that s_{j+1} is a neighbor of s_j and it is a leaf. Let s_k, s_m, and s_l be the neighbors of s_j different from s_{j+1} such that $k < j$, and $m > l > j$ (see Figure 9(a)), such that s_m is on the spine of K. Let d_1, d_2, d_3, d_4 be directions such that $d_1(s_j) = s_k$, $d_2(s_j) = s_l$, $d_3(s_j) = s_{j+1}$, and $d_4(s_j) = s_m$. Set $t^j_{d_1} = t^k_{d_1^{-1}}$ (this ensures that tile t^j can attach to the neighboring tile t^k), $t^j_{d_2} = a_{i+1}$, $t^j_{d_3} = a_{i+2}$ (hence using new binding domains for the hairs incident to s_j), and $t^j_{d_4} = \overline{t^j_{d_1}}$.

(ii) s_j is a single anchor: Let s_j be an anchor for a caterpillar K' in L_{i+1} for some i. It follows that s_j also belongs to a caterpillar K in L_i (see Figure 9(b)). Let s_k, s_m, s_l, and d_1, d_2, d_3, d_4 be defined as in case (i). Note that s_{j+1} is a neighbor of s_j and it is a leaf of K. Set $t^j_{d_1} = t^k_{d_1^{-1}}$ (this ensures that tile t^j can attach to the neighboring tile t^k), $t^j_{d_2} = a_{i+1}$, (as the caterpillar K'

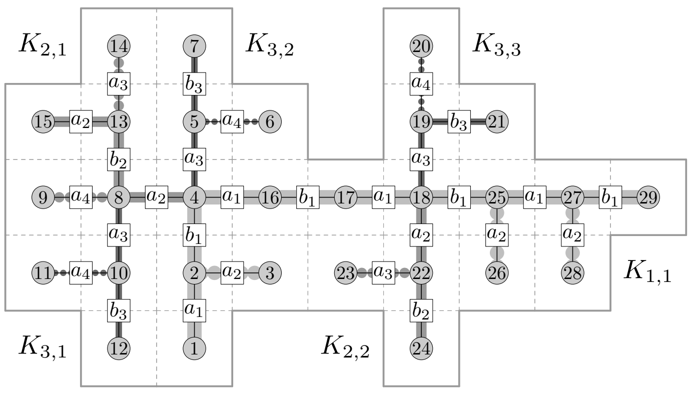


Fig. 10. The shape constructed from the tile system described in the proof of Theorem 2 based on the maximal-layer decomposition depicted in Figure 6. The circle in the center of each tile shows the tile set into which the tile belongs (and is based on the determined order of vertices). The binding domains are shown in squares.

will use a new pair of binding domains a_{i+1}, b_{i+1}), $t^j_{d_3} = a_{i+2}$ (hence using a new binding domain for the hair incident to s_j), and $t^j_{d_4} = \overline{t^j_{d_1}}$.

(iii) s_j is a double anchor: Let s_j be an anchor for a caterpillar in L_{i+1} and a caterpillar in L_{i+2}. Let s_k, s_m, s_l, and d_1, d_2, d_3, d_4 be defined as in case (i). Since the neighbor s_{j+1} of s_j is visited before the neighbor s_l, s_{j+1} has a higher layer than s_l. Therefore, s_{j+1} has layer $i+2$ and s_l has layer $i+1$. Set $t^j_{d_1} = t^k_{d_1^{-1}}$ (this ensures that tile t^j can attach to the neighboring tile t^k), $t^j_{d_2} = a_{i+1}$ (as the caterpillar in L_{i+1} will use a new pair of binding domains a_{i+1}, b_{i+1}), $t^j_{d_3} = a_{i+2}$ (as the caterpillar in L_{i+2} will use a new pair of binding domains a_{i+2}, b_{i+2}), and $t^j_{d_4} = \overline{t^j_{d_1}}$.

An example of the tile system determined by this procedure is depicted in Figure 7.

Uniqueness. Let the configuration C be defined by the map $C : \mathbb{Z} \times \mathbb{Z} \to \{t^0, t^1, \ldots, t^{|V(S)|-1}\}$ such that

$$C(v) = \begin{cases} t^j & \text{if } v = s_j \text{ for some } s_j \in V(F), \\ \mathit{empty} & \text{otherwise.} \end{cases}$$

Clearly, the configuration C has shape S. We will show that the step-wise tile assembly system $\mathbf{T}_{step} = (\Sigma, \{t^j, \mathit{empty}\}_{j=1}^{|V(S)|-1}, \{C_{t^0}\}, s_\Sigma, \{1\}_{j=1}^{|V(S)|-1})$ uniquely produces C. To this end, let $C_j = C[V_j]$, where $V_j = \{s_0, \ldots, s_j\}$, be the subconfiguration of C induced by the first $j+1$ vertices ($j = 0, \ldots, |V(S)|-1$). We say that the configuration C_j is *hierarchical*, if all of the following hold:

(H1) All exposed sides of C_j have binding domains of distinct ranks.

(H2) Let $e_1, e_2, \ldots, e_m$ be all exposed sides of C_j ordered increasingly by the ranks of their binding domains. Let $t^{i_1}, t^{i_2}, \ldots, t^{i_m}$ be the tiles in the corresponding positions. Then $i_1 \leq i_2 \leq \cdots \leq i_m$. For convenience, the rank of binding domain of exposed side e will be referred to as *the rank of e*. Using this notation, this condition can be restated as follows: the tiles with exposed sides of smaller ranks have been added before the tiles with exposed sides of higher ranks.

We will show by induction that the tile assembly system $\mathbf{T}_j$ of $\mathbf{T}_{step}$ uniquely produces C_j for each $j \geq 1$.

By definition of the step-wise tile assembly system $\mathbf{T}_{step}$, the seed configuration is $C_0 = C_{t^0}$. Since C_0 contains only one tile, condition (H2) is trivially satisfied. Since there is only one exposed side in C_0, condition (H1) is also trivially satisfied. Hence, C_0 is hierarchical. Now we show that for every $j \geq 1$, if the set of seed configurations of $\mathbf{T}_j$ is $\{C_{j-1}\}$, and C_{j-1} is hierarchical, then $\mathbf{T}_j$ produces the unique terminal assembly C_j, and C_j is hierarchical. The claim will follow.

By the induction hypothesis, the set of seed configurations of $\mathbf{T}_j$ is $\{C_{j-1}\}$, i.e., $\mathbf{T}_j = (\Sigma, \{t^j, \mathit{empty}\}, \{C_{j-1}\}, s_\Sigma, 1)$. Let the vertex s_k be the neighbor of s_j in the tree F such that $k < j$. Since $C_{j-1}(s_j) = \mathit{empty}$, by our construction t^j can attach to C_{j-1} at position s_j in step j. Since t^j is the only non-*empty* tile of the tile set used in this step, it will attach at this position. Next we show that t^j cannot attach anywhere else. Let r be the rank of the binding domain via which tiles t^j and t^k are attached to each other in our construction. Due to our assignment of binding domains, r is a lowest rank of non-*null* binding domains on t^j (there could be two binding domains on t^j with the lowest rank), and r is the highest rank of exposed sides on t^k. By the depth-first search, either $t^k = t^{j-1}$ or t^{j-1} is a leaf of F and the algorithm backtracked to t^k, in which case the tiles $t^{k+1}, \ldots, t^{j-1}$ do not have any exposed sides. This implies that the exposed side of highest rank in C_{j-1} is on t^k, since C_{j-1} is hierarchical. As r is the highest rank of exposed sides on t^k, it follows that r is the highest rank of exposed sides in C_{j-1}. Since all exposed sides in C_{j-1} have distinct rank and r is the lowest rank of non-*null* binding domains on t^j, tile t^j cannot attach anywhere else. Since all non-*null* binding domains on t^j are distinct, another copy of t^j cannot attach to the last added tile (t^j) to C_j. Hence, $\mathbf{T}_j$ uniquely produces C_j.

To complete the induction step it remains to show that C_j is hierarchical. Let $X = x_1, x_2, \ldots, x_m$ be the exposed sides in C_{j-1} ordered by increasing rank. It follows from above that, in C_j, tile t^j is attached to exposed side x_m of t^k. As before, let r be the rank of x_m. Since C_{j-1} satisfies (H1), and r is the lowest non-*null* rank of the binding domains on t^j, and the exposed sides on t^j in C_j are distinct, it follows that C_j satisfies (H1). Let $Y = y_1, y_2, \ldots, y_k$ be the exposed sides in C_j ordered by increasing rank. Since r is the highest rank of all exposed sides in C_{j-1} and r is also the lowest rank of all non-*null* binding domains on tile t^j, in the sequence Y all non-*null* binding domains of t^j are at the end of the sequence. Since all the preceding elements are a subsequence of X, they satisfy

(H2). Moreover, since all new (if any) exposed sides in Y come from tile t^j, C_j also satisfies (H2). Thus, C_j is hierarchical. This completes the induction.

Hence, the step-wise tile assembly system $\mathbf{T}_{step}$ uniquely produces the terminal assembly C, and therefore, $\mathbf{T}_{step}$ uniquely produces shape S.

Bound on the number of tile types. It remains to show that the tile complexity of $\mathbf{T}_{step}$ is as required. There are four categories of tile types: tiles with exactly one, two, three, or four non-*null* binding domains. For the first type there are two choices for binding domain type (a_i or b_i, for some i), four choices for which side of the tile will be assigned the selected binding domain, $\delta + 2$ choices for the rank of a_i, and δ choices for the rank of b_i. Hence there are $4 \cdot (\delta + 2) + 4 \cdot \delta = 2 \cdot 4 \cdot \delta + 8$ such tiles. For tiles with exactly two non-*null* binding domains, both binding domains must have the same rank, i.e., the binding domains must be a_i and b_i for some i. There are $4 \cdot 3$ ways of assigning these binding domains to the sides of a tile, and there are δ choices for i. Hence, there are $4 \cdot 3 \cdot \delta$ different tiles with exactly two non-*null* binding domains. Tiles with exactly three non-*null* binding domains have binding domains a_i, b_i, a_{i+1} for some i. There are $4 \cdot 3 \cdot 2$ ways of assigning these binding domains to the sides of a tile. There are δ choices for i. Thus, there are $4 \cdot 3 \cdot 2 \cdot \delta$ different tiles with exactly three non-*null* binding domains. Similarly, tiles with exactly four non-*null* binding domains have binding domains a_i, b_i, a_{i+1}, a_{i+2} for some i. There are $4 \cdot 3 \cdot 2$ ways of assigning these binding domains to the sides of a tile. Again, there are δ choices for i. Thus, there are $4 \cdot 3 \cdot 2 \cdot \delta$ different tiles with exactly four non-*null* binding domains. Therefore, the shape S can be assembled uniquely by a tile assembly system with tile complexity of at most $2 \cdot 4 \cdot \delta + 8 + 4 \cdot 3 \cdot \delta + 4 \cdot 3 \cdot 2 \cdot \delta + 4 \cdot 3 \cdot 2 \cdot \delta = 68\delta + 8$.

If the maximum degree of F is three, then the tile complexity of $\mathbf{T}_{step}$ is at most $2 \cdot 4 \cdot \delta + 8 + 4 \cdot 3 \cdot \delta + 4 \cdot 3 \cdot 2 \cdot \delta = 44\delta + 8$, since there are no tiles with non-*null* binding domains on all four sides used in the construction.

If the maximum degree of F is 2, then F is a path and only two non-*null* binding domains are needed: a_1 and b_1. There are 12 tile types using both a_1 and b_1 (exactly once). In addition, two tile types having only one non-*null* binding domain are used for the endpoints of the path F. Thus, if the maximum degree of F is 2, the tile complexity of $\mathbf{T}_{step}$ is 14. $\square$

To illustrate significant differences between the standard and the step-wise tile assembly models, let us consider any "path" shape, i.e., a connected shape in which every tile except two have exactly two neighbors and the remaining two tiles of the shape have exactly one neighbor each. Let n be the number of tiles of this shape. By (Adleman et al., 2002) (Theorem 4.3), in the standard model this shape requires n tile types and is produced in one step. By Theorem 2, in the step-wise model, the shape can be assembled with 14 tile types and is produced in $n - 1$ steps. Indeed, no two tile positions in a path shape are compatible (see (Adleman et al., 2002) for details), and hence, Theorem 4.3 of (Adleman et al., 2002) can be applied. On the other hand, since the degree of the spanning tree of the shape is two, Theorem 2 yields 14 tile types. The extreme example would be a "line" shape ($1 \times n$ rectangle) which uses only 4 tile types in the construction out of the 14 accounted for in the bound of Theorem 2.

The construction in the proof of Theorem 2 can be modified so that the condition (MLD3) could be omitted from the definition of the maximal-layer decomposition and still produce a tile system than uniquely assembles a given shape. However, the number of binding domains could increase. In this case, a vertex of degree 4 that is a double anchor could be an anchor for two caterpillars K and K' that are both in layer L_i for some i. To ensure the tile assembly system uniquely produces the desired shape, we cannot use binding domains a_i and b_i for the spine of both K and K'. Instead we introduce new binding domains c_i and d_i that we use for caterpillars in layer L_i. For example, for the spine of K we use the binding domains a_i and b_i and for the spine of K' we use c_i and d_i. This implies that we need at most $136\delta' + 8$ non-*null* binding domains, where δ' is the depth of this new layer decomposition. Obviously, $\delta/2 \leq \delta' \leq \delta$. In the next section in order to show a connection to a node search number we will omit condition (MLD3).

4 Layer Decompositions and Monotone Connected Node Search Number

Layer decompositions of a tree are closely related to its monotone connected node search number, which is defined later in this section. However, the definition of layer decomposition from the previous section is tailored to optimize the number of tile types used. In what follows we will show that if in a maximal-layer decomposition we do not require caterpillars to be maximal, then this does not change the maximal-layer depth of a tree. Moreover, if in addition we omit condition (MLD3) from the definition, i.e., we allow two caterpillars in the same layer to share a common vertex (anchor), the "new" layer depth corresponds to the monotone connected node search number of the tree.

Given a tree F, the sequence of sets of caterpillars $\{L_i\}_{i=1}^{\delta}$ forms a *layer decomposition* D_F of F if it satisfies all of the following:

(LD1) All caterpillars in $\bigcup_{i=1}^{\delta} L_i$ are edge disjoint, and cover all edges of F.
(LD2) L_1 consists of a single natural caterpillar of F.
(LD3) For every $2 \leq i \leq \delta$, caterpillars in L_i are vertex disjoint.
(LD4) For every $2 \leq i \leq \delta$, every caterpillar K in L_i is a natural caterpillar anchored at a vertex v on the spine of some caterpillar in L_{i-1}.

Observe that this definition differs from the definition of maximal-layer decomposition only that we do not require the caterpillars to be maximal in (LD2) and (LD4). An example of a layer decomposition is shown in Figure 11. As before, the *depth* of a layer decomposition $D_F = \{L_i\}_{i=1}^{\delta}$ is δ. The *layer depth of a tree* F, denoted LD-depth(F), is the smallest δ such that F has a layer decomposition of depth δ. We will show that these changes to the definition of layer decomposition do not affect the layer depth of a tree.

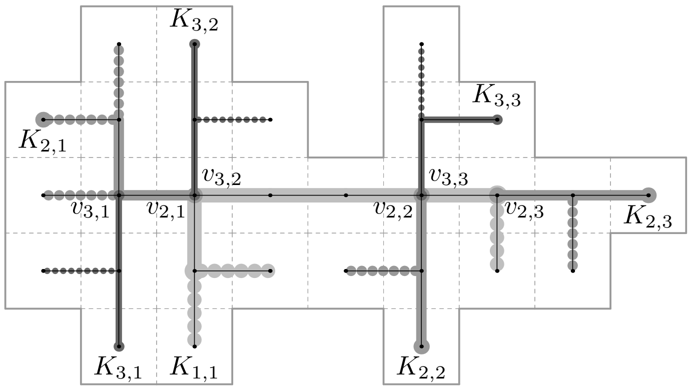


Fig. 11. An example of a layer decomposition of a tree spanning the given shape with layers $L_1 = \{K_{1,1}\}$, $L_2 = \{K_{2,1}, K_{2,2}, K_{2,3}\}$ and $L_3 = \{K_{3,1}, K_{3,2}, K_{3,3}\}$. The anchor of a caterpillar $K_{i,j}$ is denoted as $v_{i,j}$.

Lemma 1. *For any tree F,* MLD-depth(F) = LD-depth(F).

Proof. Since any maximal-layer decomposition of F is also a layer decomposition of F, it follows that LD-depth$(F) \leq$ MLD-depth(F). On the other hand, let $D_F = \{L_i\}_{i=1}^{\delta}$, where $\delta =$ LD-depth(F), be a layer decomposition of F with the smallest number of caterpillars. We argue that D_F is also a maximal-layer decomposition of F.

Towards a contradiction, suppose that D_F is not a maximal-layer decomposition, i.e., either the caterpillar in L_1 is not maximal, or a caterpillar in a higher layer set L_i, $i > 1$, is not maximal anchored at a vertex u on the spine of a caterpillar in L_{i-1}. Let this caterpillar be K. In both cases, let v be the endpoint of the spine of K different from the anchor u of K. Since K is not maximal, v is not a leaf of F. Hence, by (LD1) and (LD4), there is another caterpillar K' in L_{i+1} which is anchored at v. Let us modify D_F as follows: First, append K' to K at v, and leave the resulting caterpillar in L_i. Second, remove K' from L_{i+1}. Let F_u be the tree F rooted at u, and $F_u[v]$ be the subtree of F_u rooted at v. By (LD1), every caterpillar of D_F is either completely in $F_u[v]$ or has no edges in common with $F_u[v]$. Third, move every caterpillar in $F_u[v]$ from its layer set L_j to L_{j-1}.

This modification obviously does not violate (LD1) and (LD2). Before our modification, every caterpillar in $F_u[v]$ except K' was in a layer greater than $i + 1$. Since K and K' have been concatenated and since all caterpillars in $F_u[v]$ decreased their layers consistently, (LD3) and (LD4) also hold. Hence, the new decomposition is a layer decomposition and it is easy to see that its depth is at most δ. This is a contradiction, since the new decomposition has one less caterpillar than D_F. Hence D_F is a maximal-layer decomposition and MLD-depth$(F) \leq$ LD-depth(F). Therefore, MLD-depth(F) = LD-depth(F).

$\square$

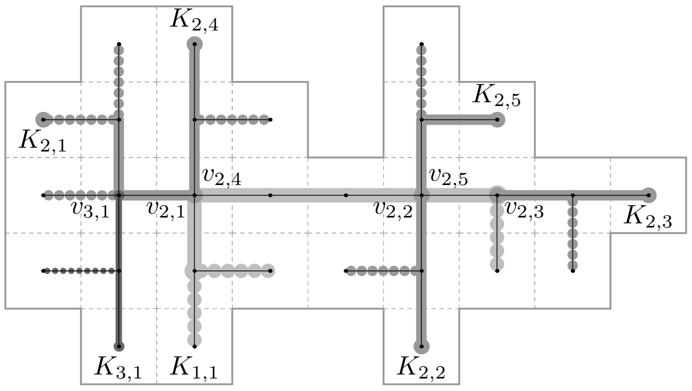


Fig. 12. An example of a general layer decomposition of a tree spanning the given shape with layers $L_1 = \{K_{1,1}\}$, $L_2 = \{K_{2,1}, K_{2,2}, K_{2,3}, K_{2,4}, K_{2,5}\}$ and $L_3 = \{K_{3,1}\}$. The anchor of a caterpillar $K_{i,j}$ is denoted as $v_{i,j}$.

As we noted in Section 3, condition (LD3) is used only to optimize tile complexity. In order to achieve equivalence between the layer depth of a tree and its monotone connected node search number we omit this condition and obtain a "general layer decomposition".

Given a tree F, the sequence of sets of caterpillars $\{L_i\}_{i=1}^{\delta}$ forms a *general layer decomposition* of F if it satisfies all of the conditions: (LD1), (LD2), and (LD4). An example of a general layer decomposition is shown in Figure 12.

The *depth* of a general layer decomposition $\{L_i\}_{i=1}^{\delta}$ is again δ, the number of sets in the decomposition. The *GLD-depth of a tree F*, denoted GLD-depth(F), is the smallest δ such that F has a general layer decomposition of depth δ. The *GLD-depth of a shape S* is the minimum GLD-depth of all spanning trees of S. The following observation easily follows.

Observation 3 *For any tree F,*

$$(\text{LD-depth}(F) + 1)/2 \leq \text{GLD-depth}(F) \leq \text{LD-depth}(F)\,.$$

Proof. The second inequality follows from the fact that the layer decomposition is also a general layer decomposition. For the first inequality, consider a general layer decomposition D_F of F with δ layers. For a caterpillar K in D_F anchored on caterpillar K' in D_F, we say that K is a child of K'. Now, we define the relation on caterpillars in D_F "is a descendant" as the transitive closure of the relation "is a child". Next, we describe the process how to transform D_F into a layer decomposition D'_F which at most doubles the number of layers. The process will move some caterpillars to higher layers. The caterpillar in layer 1 stays in layer 1. Assume that caterpillars in layers $1, \ldots, i$ satisfy condition (LD3). Assume that two caterpillars K_1, K_2 in layer $i + 1$ are not disjoint, i.e., they are anchored at the same vertex of a caterpillar in layer i. Keep caterpillar K_1 in layer $i + 1$, and move caterpillar K_2 and all its descendants one layer up. We do the same for all non-disjoint pairs of caterpillars in layer $i+1$. Note that this may increase the depth by at most one. Since for each layer other than the first, we increase the depth by at most one, the constructed layer decomposition has depth at most $2\delta - 1$, and the first inequality follows. □

Node searching is a variant of graph searching which was first introduced by Kirousis and Papadimitriou (1986). We are given a graph whose edges are all "contaminated", and a set of searchers. The goal is to obtain a state of the graph in which all edges are simultaneously "cleared". In node searching, an edge becomes *cleared* if both its endpoints are concurrently occupied by searchers. An edge e becomes recontaminated, if there is a path from a contaminated edge to e and there is no searcher on this path. In node searching there are two basic operations, called "search steps": (i) place a searcher on a vertex, and (ii) remove a searcher from a vertex.

A *search strategy* is a sequence of search steps that results in all edges of the graph being simultaneously cleared. The smallest number of searchers for which a search strategy exists for a graph G is called the *node search number* $ns(G)$ of G. A search strategy is *monotone* if no recontamination ever occurs. A search strategy is *connected* if the set of cleared edges always induces a connected subgraph. We are interested in search strategies that are both monotone and connected, and call the minimum number of searchers for which such a strategy exists for the graph G the *monotone connected node search number*, denoted by $mcns(G)$.

Edge searching is a version of graph searching where, in addition to the two search steps allowed in node searching, searchers can slide along edges. Moreover, a necessary condition for an an edge uv to become cleared is that a searcher slides along the edge from endpoint u to endpoint v. The edge search number of a graph G is simply denoted by $s(G)$. Barriere et al. (2003) characterized the trees T whose monotone connected (edge) search number, $mcs(T)$, is at most k, for a given integer k. This characterization is in terms of k-caterpillars, which are similar to our layer-decompositions. Furthermore, it was shown in (Barriere et al., 2003) that there is a unique obstruction (a forbidden minor subgraph) for the class of trees T such that $mcs(T) \leq k$. These results can easily be adapted to layer-decompositions and monotone connected node search. To state the result, we first need the following definitions.

Let B'_k be the rooted tree obtained from the complete binary tree of depth k (the distance from the root to the leaves) by appending a new vertex v_i to each leaf l_i. Let D'_k be the rooted tree obtained by joining the three roots of three copies of B'_{k-1} to a new root r (see Figure 13). We say a graph H is a (tree) *minor* of a graph G, denoted $H \preceq G$, if H is isomorphic to a graph obtained from a subgraph of G by zero or more edge contractions (removing an edge and identifying its endpoints).

Theorem 4. *For any tree F, the following three properties are equivalent:*

1. GLD-depth$(F) \geq k$;
2. $D'_{k-1} \preceq F$;

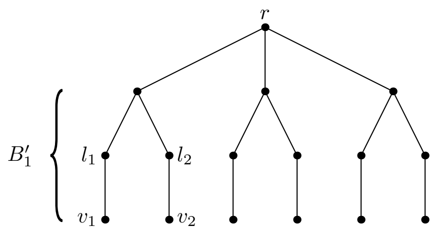


Fig. 13. The tree D'_2.

3. $mcns(F) \geq k+1$.

Proof. We prove the theorem in a similar way as the proof of Theorem 4.1 in (Barriere et al., 2003). We will use the following lemmas which we will prove later.

Lemma 2. *For any* $k \geq 1$, $mcns(D'_k) \geq k+2$.

Lemma 3. *For any tree* F, *if* GLD-depth$(F) \leq k$ *then* $mcns(F) \leq k+1$.

Lemma 4. *For any tree* F, *if* $D'_k \not\preceq F$ *then* GLD-depth$(F) \leq k$.

The claim now follows directly from these lemmas. If GLD-depth$(F) \geq k > k-1$, then by Lemma 4, we have $D'_{k-1} \preceq F$. Now suppose $D'_{k-1} \preceq F$. By Lemma 2, $mcns(D'_{k-1}) \geq k+1$, thus also $mcns(F) \geq k+1$. Finally, if $mcns(F) \geq k+1 > k$, then by Lemma 3, we have GLD-depth$(F) > k-1$. Hence, GLD-depth$(F) \geq k$. □

It remains to prove the lemmas.

Proof (Proof of Lemma 2). We will show that for any monotone connected node search strategy in D'_k, there is a step in which at least $k+2$ searchers are required to avoid recontamination. To this end, let T_1, T_2, and T_3 be the three subtrees attached to the root r of D'_k. Note that these subtrees are isomorphic to B'_{k-1}. Let i_1 be the first step during which an edge incident to the root r is cleared by a searcher. Since we are performing a connected search strategy, without loss of generality, we may assume that the search starts in T_3 or in the root, and hence, T_1 and T_2 are still completely contaminated at step i_1. Let i_2 be the first step during which and edge e incident to a leaf l of T_1 or T_2 is cleared by a searcher. We may assume l belongs to T_1. At this point edges incident to all other leaves of T_1 and all leaves of T_2 are still contaminated, while the path P from the leaf l to the root r is cleared (monotone connected search strategy). Searchers are located at the leaf l and its neighbor n (which is also a vertex of P), as otherwise edge e could not have been cleared during step i_2. For every other vertex x on the path P (including r), there is a path from x to a contaminated leaf. Since these paths are disjoint, each path needs to be guarded by a searcher. Thus, at least $k+2$ searchers are required during step i_2. Hence, $mcns(D'_k) \geq k+2$. □

Proof (Proof of Lemma 3). Let $\{L_i\}_{i=1}^k$ be a general layer decomposition of F of depth k. Let the path P be the spine of the caterpillar in L_1. We show that there is a monotone connected node search strategy using $k+1$ searchers starting at one endpoint of P. Informally, we will have one dedicated searcher for each layer and one extra searcher for clearing all edges of F. The search starts at an endpoint of the spine of the caterpillar in layer 1 and continues along its spine. When visiting a next vertex of the spine of a caterpillar in layer i, we leave the i-th searcher at this vertex and use the extra searcher to clear leaves. If this vertex is an anchor of a caterpillar (or two caterpillars) in layer $i+1$, inductively, we clear the subtree induced by each caterpillar and all its descendants using the searchers $i+1$ up to k and the extra searcher. After cleaning this subtree, only searchers 1 to i are in use, and the search continues along the spine of the caterpillar in layer i. Formally, we proceed by induction on k.

For the base case of $k = 1$, F is itself a caterpillar. Clearly, there exists a monotone connected node search strategy using 2 searchers starting at one endpoint of the spine of F. For the induction hypothesis, suppose that for every general layer decomposition $\{L'_i\}_{i=1}^{\delta}$ of a tree F' of depth at most $k-1$, there exists a monotone connected node search strategy starting at one endpoint of the spine of the caterpillar in L'_1, using at most $\delta + 1 \leq k$ searchers.

For the induction step, let F be a tree with a general layer decomposition $\{L_i\}_{i=1}^k$ of depth k. Let $P = \{v_0, v_1, \ldots, v_m\}$ be the spine of the caterpillar in L_1. For each vertex v_i denote by $w_{i,0}, w_{i,1}, \ldots, w_{i,d_i}$ the set of neighbors of v_i that are not in P. Denote by F_{v_i} the tree F rooted at v_i. Let $F_{v_i}[w_{i,j}]$ be the subtree of F_{v_i} rooted at $w_{i,j}$. Note that the tree $F_{v_i}[w_{i,j}]$ together with the edge $v_i w_{i,j}$, denoted $F_{v_i}[w_{i,j}] + v_i w_{i,j}$, is a tree that has a general layer decomposition of depth at most $k-1$ where v_i is an endpoint of the caterpillar in the first set of this decomposition (for example, take the general layer decomposition "induced" by $\{L_i\}_{i=1}^k$, i.e., $\{L_i \cap V(F_{v_i}[w_{i,j}] + v_i w_{i,j})\}_{i=2}^k$). We now use the following search strategy for F: place all $k+1$ searchers at v_0. Every time all searchers reach a new vertex v_i of P, leave one searcher at v_i and use k searchers to perform a monotone connected node search (starting from v_i) on the subtree $F_{v_i}[w_{i,j}] \cup v_i w_{i,j}$, for every $j = 0, \ldots, d_i$. Then move one searcher to v_{i+1} and leave the remaining k searchers at v_i to clear the edge $v_i v_{i+1}$. Now move the remaining searchers from v_i to v_{i+1}. This is a monotone connected node search strategy using $k+1$ searchers. Hence, $mcns(F) \leq k+1$. □

Proof (Proof of Lemma 4). We first need the following definition. Given two trees F_1 and F_2 with roots x_1 and x_2, respectively, we say F_1 *is a x_2-rooted minor of* F_2, denoted by $F_1 \preceq_{x_2} F_2$, if F_1 is a minor of F_2 and vertex x_1 is either x_2 or the result of contracting a series of edges, some of which contain x_2 as an endpoint. As in the proof above, let $F_v[w]$ denote the subtree of tree F_v rooted at w, where F_v is the tree F rooted at v.

Let F be a tree. We will first show, by induction on k, that if there is a vertex v of F such that $B'_k \not\preceq_v F$, then there exists a general layer decomposition with layer depth at most k such that v is an endpoint of the spine of the caterpillar in layer one of this decomposition. Then we show that for any tree F such that $D'_k \not\preceq F$, there exists a vertex v such that $B'_k \not\preceq_v F$.

For the base case of $k = 1$, assume that for v of F, $B'_1 \not\preceq_v F$. Now, for any vertex w of F_v, at most one child of w is not a leaf. Hence, F is a caterpillar and we can choose its spine so that v is one of its endpoints. Therefore, GLD-depth$(F) = 1$.

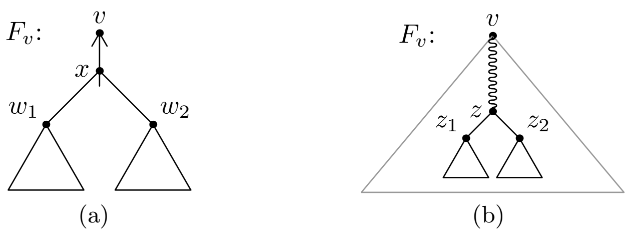


Fig. 14. (a) The case when not all vertices of S lie on a path starting at v, i.e., there are w_1 and w_2 in S lying on two different paths starting at v. Since $B'_{k-1} \preceq_{w_1} F_v[w_1]$ and $B'_{k-1} \preceq_{w_2} F_v[w_2]$, $B'_k \preceq_x F_v[x]$, where x is the lowest common ancestor of w_1 and w_2. This implies that also $B'_k \preceq_v F_v$, which is a contradiction. (b) Two subtrees of F_z, $F_z[z_1]$ and $F_z[z_2]$ with minor B'_{k-1}.

Now suppose the claim holds for some $k - 1 \geq 1$, and there is a vertex v such that $B'_k \not\preceq_v F$. We consider two cases.
Case $B'_{k-1} \not\preceq_v F$. By the induction hypothesis, there exists a general layer decomposition with layer depth at most $k - 1 < k$ such that v is an endpoint of the spine of the layer one caterpillar.
Case $B'_{k-1} \preceq_v F$. Let S be the set consisting of all vertices w such that $B'_{k-1} \preceq_w F_v[w]$. Since $B'_k \not\preceq_v F$, the vertices in S induce a path starting at v in F, or otherwise, we have the situation depicted in Figure 14(a). Now we can construct a general layer decomposition of F as follows: The first layer contains the caterpillar with the spine induced by vertices in S and all leaves of F which are not in S and are adjacent to this spine are hairtips of this caterpillar. Now, consider a vertex z not in this caterpillar but adjacent to a vertex w of this caterpillar (a vertex in S). By induction hypothesis, there exists a general layer decomposition with depth at most $k - 1$ such that z is an endpoint of the spine of the layer one caterpillar K in this decomposition. We include this sub-decomposition into decomposition F by extending the spine of K with the vertex w and increasing layer number by one for each caterpillar in this sub-decomposition. The vertex w becomes the anchor of K in the constructed decomposition of F. We repeat this process for such z. It follows that the layer depth of the constructed decomposition is at most k and v is an endpoint of the spine of the layer one caterpillar.

It remains to show that if $D'_k \not\preceq F$, then there exists a vertex v such that $B'_k \not\preceq_v F$. Towards a contradiction, suppose that for every vertex v of F, $B'_k \preceq_v$

F. We will show that then $D'_k \preceq F$. To show that it is enough to find two disjoint rooted subtrees of F_z, for some z of F, such that B'_{k-1} is a rooted minor of one subtree and B'_k is a rooted minor of the other subtree of F_z. Indeed, in this case, D'_k is a rooted minor of the subtree rooted in the lowest common ancestor of the two subtrees. Pick any v of F. Since $B'_k \preceq_v F$, there is a z with children z_1 and z_2 in F such that $B'_{k-1} \preceq_{z_1} F_z[z_1]$ and $B'_{k-1} \preceq_{z_2} F_z[z_2]$, cf. Figure 14(b). Since $B'_k \preceq_{z_1} F$, either $B'_k \preceq_{z_1} F_z[z_1]$ or $B'_k \preceq_z F_{z_1}[z]$. Note that the subtrees $F_z[z_1]$ and $F_{z_1}[z]$ and the edge $\{z_1, z\}$ form a decomposition of F. In the first case, B'_k is a rooted minor of $F_z[z_1]$ and B'_{k-1} is a rooted minor of $F_z[z_2]$, and hence, $D'_k \preceq F$, a contradiction. In the second case, B'_{k-1} is a rooted minor of $F_z[z_1]$ and B'_k is a rooted minor of $F_{z_1}[z]$, and hence, again, $D'_k \preceq F$, a contradiction. □

Rephrasing the equivalence of properties (1) and (3) we get:

Corollary 1. *For any tree F, $mcns(F) = \text{GLD-depth}(F) + 1$.*

In (Fraigniaud and Nisse, 2008) it was shown that the connected node search number of a graph is not always equal to the monotone connected node search number, but for trees these two search numbers are equal. In (Best, 2010), it was proved that the connected node search number is one more than the **connected pathwidth**, cf. (Best, 2010) for the definition. This implies that the GLD-depth of F is equal to the connected pathwidth of F.

5 Shapes with Constant Tile Complexity

A *rectangle* R is a shape for which there exist integers $N \geq 2$ and $M \geq 2$ and a vertex (x_0, y_0) such that vertex $(x, y) \in V(R)$ if and only if $x_0 \leq x < x_0 + N$ and $y_0 \leq y < y_0 + M$. We say that two disjoint rectangles R_1 and R_2 are *joined* by an edge $e = uv$ of the square lattice if $u \in V(R_1)$ and $v \in V(R_2)$. Given a shape S and an integer $k \geq 1$, we say S has a *rectangle decomposition* $\{R_i\}_{i=1}^m$ *with connectors of size* k if there exist m disjoint rectangles, $R_1, \ldots, R_m$, such that $V(S) = \bigcup_{i=1}^m V(R_i)$ and for each $i > 1$, there exists $j < i$ such that the rectangles R_i and R_j are joined by at least k disjoint edges. A selected set of k such edges whose endpoints induce a rectangle is called a *connector of size* k of R_i and R_j. Note that the endpoints of this connector which are in R_i (respectively, in R_j) induce a path of length $k-1$ which is called an *interface of* R_i (respectively, R_j). Observe that since the rectangles in the rectangle decomposition are disjoint, interfaces of each rectangle are disjoint. For an example of three disjoint rectangles with connectors of size 2 see Figure 15.

Theorem 5. *A shape S which has a rectangle decomposition $\mathbf{R} = \{R_i\}_{i=1}^m$ with connectors of size 2, can be assembled using at most 24 non-empty tile types.*

Proof. We will show that S has a spanning tree F of maximum degree at most 3, maximal-layer depth at most 2, and a very specific shape (all caterpillar in the second layer are horizontal paths). Given this spanning tree, we will obtain a

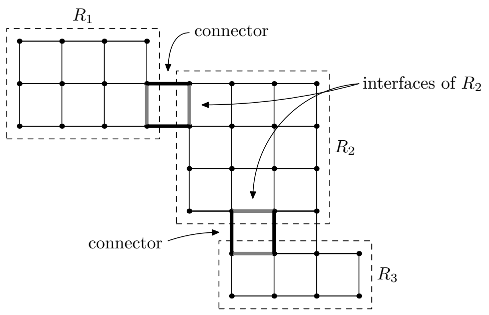


Fig. 15. A rectangle decomposition of a shape with connectors of size 2. Connector edges are depicted in bold and interface edges are depicted in grey.

tile assembly system following the construction in the proof of Theorem 2. Due to specific shape of F, this tile assembly system will only use 24 non-*empty* tile types which is slightly better than the general bound (96) given by the theorem.

To obtain a suitable spanning tree F of S we will first show that S has a spanning subgraph G satisfying all of the following conditions:

(i) The maximum degree of G is 3.
(ii) G contains a cycle C that consists of all connector edges and all boundary edges of each rectangle in $\mathbf{R}$ which are not interface edges.
(iii) Every vertex not on the cycle C is on a "horizontal" path whose west endpoint belongs to C.

We will show this by induction on the number of rectangles in the rectangle decomposition $\mathbf{R}$ of S.

For the base case when $m = 1$, S is itself a rectangle. Let C be the cycle consisting of all boundary edges of S. For every vertex w on the western boundary of S except the two corner vertices, let P_w be the path starting at w and going east with the other endpoint being the last vertex that is not on the boundary of S (see Figure 16). Then the cycle C together with all paths P_w forms a desired spanning subgraph G of S.

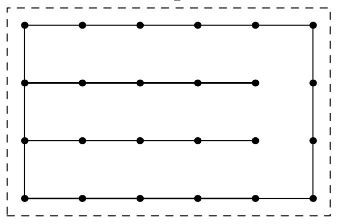


Fig. 16. The spanning subgraph for the case of a single rectangle.

For the induction step, let S be any shape which has a rectangle decomposition $\mathbf{R} = \{R_i\}_{i=1}^{m}$ with connectors of size 2, for some integer $m \geq 2$. Let S' be the shape corresponding to the rectangle decomposition $\{R_i\}_{i=1}^{m-1}$. Note that by definition of rectangle decomposition S' is indeed a shape (a connected induced subgraph of the lattice). By induction hypothesis, S' has a spanning subgraph G' satisfying conditions (i)–(iii). Since $\mathbf{R}$ has connectors of size 2 there exist two adjacent vertices u_m and v_m on the boundary of R_m such that for some $i < m$, u_m is adjacent to a vertex u_i of R_i and v_m is adjacent to a vertex v_i of R_i. Since u_i and v_i are on the boundary of R_i and are not in any interface of S', by (ii), the edge u_iv_i is on cycle C' of G'. Obtain a spanning subgraph G_m of R_m as described in the base case. Let C_m be the cycle of G_m. To obtain the cycle C for the spanning subgraph G of S, join C' and C_m by adding the connector edges u_iu_m and v_iv_m, and deleting the interface edges u_iv_i and u_mv_m (see Figure 17). Let G be the spanning subgraph of S obtained by the union of C and the "horizontal" paths of G' and G_m.

We will show that G satisfies all three conditions. Since for every vertex $v \in G$, either $\deg_G(v) = \deg_{G'}(v)$ or $\deg_G(v) = \deg_{G_m}(v)$, and by the induction hypothesis, the maximum degree of G' and G_m is three, also the maximum degree of G is three (condition (i)). By the induction hypothesis, both G' and G_m satisfy condition (ii) for the rectangle decompositions $\{R_i\}_{i=1}^{m-1}$ and R_m, respectively. Since C contains the connector edges u_iu_m and v_iv_m, and all edges of C' and C_m except the edges u_iv_i and u_mv_m which are interface edges, also G satisfies condition (ii). By our construction, every vertex of G that is not on C is on a horizontal path P. This path is either in G' or G_m and thus, by the induction hypothesis, the west endpoint of P belongs to either C' or C_m. Since all vertices of C' and C_m belong to C, the west endpoint of P also belongs to C. Hence, G satisfies condition (iii).

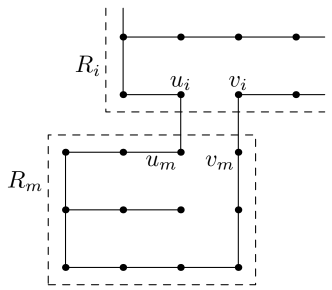


Fig. 17. Extending the spanning subgraph to include R_m.

Given this spanning subgraph G of S, we obtain a spanning tree F of S by deleting an edge from the cycle C that is on the northern boundary of S. Certainly F will still have maximum degree 3. It remains to show that F has the maximal-layer depth at most 2. Let the caterpillar K consist the spine which is the path in F that we obtained by deleting the edge from the cycle C, and hairs which are all "horizontal" paths of length 1. Let $L_1 = \{K\}$. If $K = F$, then L_1 trivially forms a layer decomposition of F. Otherwise let L_2 be the set consisting of all remaining "horizontal" paths. All caterpillars in $L_1 \cup L_2$ are edge-disjoint and cover all edges of F. Hence, $\{L_1, L_2\}$ satisfies condition (MLD1). Let u and v be the endpoints of the spine of K. Since u and v are on the northern boundary of S, they are leafs of F (any branching occurs on the western boundaries of rectangles in S). Thus K is a maximal caterpillar of F, and hence $\{L_1, L_2\}$ satisfies condition (MLD2). By our construction, all "horizontal" paths are vertex disjoint. Thus, $\{L_1, L_2\}$ satisfies condition (MLD3). Every "horizontal" path is a maximal caterpillar anchored at w, where w is an internal vertex of the spine of caterpillar K in L_1. Hence, $\{L_1, L_2\}$ satisfies condition (MLD4). Therefore, $\{L_1, L_2\}$ is a maximal-layer decomposition of F, and hence F has depth at most 2.

Now construct a tile assembly system from the spanning tree F of S as described in the proof of Theorem 2. However, due to special properties of F this tile assembly system will use less than the 96 tile types that are guaranteed by Theorem 2. In the following we will count the number of tile types which can occur in the constructed tile assembly system for F. Our tile assembly system will use $4 \cdot 3$ tile types that use both a_1 and b_1 (exactly once) and no other non-*null* binding domains. There will be six tile types that use exactly three non-*null* binding domains (a_1, b_1, and a_2) (this is because any vertices of degree 3 are on the western boundaries of rectangles in S). There are two tile types that use both a_2 and b_2 (exactly once) and no other non-*null* binding domains, because all the paths in L_2 are horizontal. For the same reason, there are only two tile types with only one non-*null* binding domain which is taken from $\{a_2, b_2\}$. Furthermore, there will be two more tile types corresponding to the endpoints of the spine of the caterpillar of L_1 (both of these tiles have only one non-*null* binding domain - either a_1 or b_1). Therefore, our tile assembly system uses only 24 non-empty tiles. □

If we are allowed to scale shapes by a factor of 2, then any shape can be assembled using a constant number of tile types.

Theorem 6. *Given an arbitrary shape S, let S' be the shape obtained by scaling S by a factor of 2. Then S' can be uniquely produced by a step-wise tile assembly system at temperature 1 using at most 14 non-empty tile types.*

Proof. Each vertex of S corresponds to a 2×2 rectangle in S'. These rectangles form a rectangle decomposition of S' with connectors of size 2. If we follow the proof of Theorem 5 to obtain a spanning tree of S', we can see that this spanning tree is in fact a Hamiltonian path (since 2×2 rectangles have no internal vertices). This also follows from Lemma 3.4 in (Summers, to appear). Thus, Theorem 2 implies that 14 non-empty tile types and 2 non-*null* binding domains suffice to uniquely assemble S'. □

6 Conclusions

In this paper, we have formally studied the step-wise tile assembly model of (Reif, 1999). A similar model, called the *staged assembly model*, has been proposed in (Demaine et al., 2008). We gave an upper bound on the number of tile types required to uniquely assemble an arbitrary shape depending on the depth of the maximal-layer decomposition of the shape, if we do not require the full connectivity. We showed that this bound is related to the monotone connected node search number (cf. (Kirousis and Papadimitriou, 1985)) of the underlying spanning tree. Further, we described a large class of shapes that have constant tile complexity (24 tile types suffice). This class includes all shapes scaled by a factor of 2. We also showed that for these scaled shapes (by a factor of 2), 14 tile types suffice. In the staged assembly model, arbitrary shape S can be constructed with a constant number of tile types (52) provided that $\log(|V(S)|)$ bins are used (Demaine et al., 2008). Note that the number of steps in our construction is linear in the size of the assembled shape and the construction from (Demaine et al., 2008) the number is steps is the diameter of the shape. In the future we would like to find a trade-off between the number of steps and the number of tile types in the step-wise tile assembly model.

Among other interesting open problems we mention the following.

1. From our proofs it is apparent that there is a freedom in the choice of the spanning tree of the target shape which can affect the number of tile types. It is an interesting question how to choose the optimal spanning tree.
2. Is there an efficient algorithm that computes the maximal-layer depth of a given tree?

Bibliography

H. Abelson, D. Allen, D. Coore, C. Hanson, G. Homsy, T. F. Knight, R. Nagpal, E. Rauch, G. J. Sussman, and R. Weiss. Amorphous computing. *Communications of the ACM*, 43:74–82, 2000.

L. Adleman, Q. Cheng, A. Goel, M.-D. Huang, D. Kempe, P. M. de Espanes, and P. Rothemund. Combinatorial optimization problems in self-assembly. In *Proceedings of STOC*, pages 23–32, 2002.

G. Aggarwal, Q. Cheng, M. H. Goldwasser, M.-Y. Kao, P. M. Espanes, and R. T. Schweller. Complexities for generalized models of self-assembly. *SIAM J. Comput.*, 34(6):1493–1515, 2005.

L. Barriere, P. Fraigniaud, N. Santoro, and D. M. Thilikos. Connected and internal graph searching. In *In 29th Workshop on Graph Theoretic Concepts in Computer Science (WG), Springer-Verlag, LNCS 2880*, pages 34–45, 2003.

M. J. Best. A bound on connected pathwidth. (manuscript), 2010.

E. D. Demaine, M. L. Demaine, S. P. Fekete, M. Ishaque, E. Rafalin, R. T. Schweller, and D. L. Souvaine. Staged self-assembly: nanomanufacture of arbitrary shapes with $O(1)$ glues. *Natural Computing*, 7(3):347–370, 2008.

P. Fraigniaud and N. Nisse. Monotony properties of connected visible graph searching. *Information and Computation*, 206(12):1383–1393, 2008.

M. Gomez-Lopez, J. Preece, and J. Stoddart. The art and science of self-assembling molecular machines. *Nanotechnology*, 7:183 – 192, 1996.

L. Kirousis and C. Papadimitriou. Searching and pebbling. *Theor. Comput. Sci.*, 47(2):205–218, 1986.

L. M. Kirousis and C. H. Papadimitriou. Interval graphs and searching. *Discrete Mathematics*, 55(1):181–184, 1985.

T. LaBean, H. Yan, J. Kopatsch, F. Liu, E. Winfree, J. H. Reif, and N. Seeman. Construction, analysis, ligation, and self-assembly of DNA triple crossover complexes. *Journal of the American Chemical Society*, 122:1848–1860, 2000.

C. Mao, T. H. LaBean, J. Reif, and N. Seeman. Logical computation using algorithmic self-assembly of DNA triple-crossover molecules. *Nature*, 407:493–496, 2000.

J. Maňuch, L. Stacho, and C. Stoll. Step-assembly with a constant number of tile types. In *Proc. of of the 20th International Symposium on Algorithms and Computation (ISAAC, Hawaii, 2009)*, number 5878 in LNCS, pages 954–963, 2009.

J. H. Reif. Local parallel biomolecular computing. In *DNA Based Computers III, volume 48 of DIMACS*, pages 217–254. American Mathematical Society, 1999.

P. Rothemund, N. Papadakis, and E. Winfree. Algorithmic self-assembly of DNA Sierpinski triangles. *PLoS Biology*, 2:2041–2053, 2004.

P. W. K. Rothemund and E. Winfree. The program-size complexity of self-assembled squares. In *Proceedings of STOC*, pages 459–468, 2000.

N. Seeman. DNA nanotechnology: novel DNA constructions. *Annual Review of Biophysics and Biomolecular Structure*, 27:225–248, 1998.

S. M. Summers. Reducing tile complexity for the self-assembly of scaled shapes through temperature programming. *Algorithmica*, to appear.

E. Winfree, X. Yang, and N. Seeman. Universal computation via self-assembly of DNA: Some theory and experiments. In *Proceedings of the Second Annual Meeting on DNA Based Computers*, pages 191–214, 1996.

E. Winfree, F. Liu, L. A. Wenzler, and N. C. Seeman. Design and self-assembly of two dimensional DNA crystals. *Nature*, 394:539–544, 1998.