

Tilt Assembly: Algorithms for Micro-Factories That Build Objects with Uniform External Forces

Aaron T. Becker^{*1}, Sándor P. Fekete², Phillip Keldenich², Dominik Krupke², Christian Rieck², Christian Scheffer², and Arne Schmidt²

¹Department of Electrical and Computer Engineering, University of Houston, USA.
atbecker@uh.edu

²Department of Computer Science, TU Braunschweig, Germany. {s.fekete, p.keldenich, d.krupke, c.rieck, c.scheffer, arne.schmidt}@tu-bs.de

Abstract

We present algorithmic results for the parallel assembly of many micro-scale objects in two and three dimensions from tiny particles, which has been proposed in the context of programmable matter and self-assembly for building high-yield micro-factories. The underlying model has particles moving under the influence of uniform external forces until they hit an obstacle. Particles bond when forced together with another appropriate particle.

Due to the physical and geometric constraints, not all shapes can be built in this manner; this gives rise to the TILT ASSEMBLY PROBLEM (TAP) of deciding constructibility. For simply-connected polyominoes P in 2D consisting of N unit-squares (“tiles”), we prove that TAP can be decided in $O(N \log N)$ time. For the optimization variant MAXTAP (in which the objective is to construct a subshape of maximum possible size), we show *polyAPX*-hardness: unless $P=NP$, MAXTAP cannot be approximated within a factor of $\Omega(N^{\frac{1}{3}})$; for tree-shaped structures, we give an $\Omega(N^{\frac{1}{2}})$ -approximation algorithm. For the efficiency of the assembly process itself, we show that any constructible shape allows *pipelined* assembly, which produces copies of P in $O(1)$ amortized time, i.e., N copies of P in $O(N)$ time steps. These considerations can be extended to three-dimensional objects: For the class of polycubes P we prove that it is *NP*-hard to decide whether it is possible to construct a path between two points of P ; it is also *NP*-hard to decide constructibility of a polycube P . Moreover, it is *expAPX*-hard to maximize a sequentially constructible path from a given start point.

1 Introduction

In recent years, progress on flexible construction at micro- and nano-scale has given rise to a large set of challenges that deal with algorithmic aspects of programmable matter. Examples of cutting-edge application areas with a strong algorithmic flavor include self-assembling systems, in which chemical and biological substances such as DNA are designed to form predetermined shapes or carry out massively parallel computations; and swarm robotics, in which complex tasks are achieved through the local interactions of robots with severely limited individual capabilities, including micro- and nano-robots.

Moving individual particles to their appropriate attachment locations when assembling a shape is difficult because the small size of the particles limits the amount of onboard energy and computation. One successful approach to dealing with this challenge is to use molecular diffusion in combination with cleverly designed sets of possible connections: in *DNA tile self-assembly*, the particles are equipped with sophisticated bonds

^{*}Work from this author was partially supported by National Science Foundation IIS-1553063 and IIS-1619278

38 that ensure only a predefined shape is produced when mixing together a set of tiles, see [31]. The resulting
 39 study of algorithmic tile self-assembly has given rise to an extremely powerful framework and produced a
 40 wide range of impressive results. However, the required properties of the building material (which must be
 41 specifically designed and finely tuned for each particular shape) in combination with the construction process
 42 (which is left to chemical reactions, so it cannot be controlled or stopped until it has run its course) make
 43 DNA self-assembly unsuitable for some applications.

44 An alternative method for controlling the eventual position of particles is to apply a uniform external
 45 force, causing all particles to move in a given direction until they hit an obstacle or another blocked particle.
 46 As two of us (Becker and Fekete, [4]) have shown in the past, combining this approach with custom-made
 47 obstacles (instead of custom-made particles) allows complex rearrangements of particles, even in grid-like
 48 environments with axis-parallel motion. The appeal of this approach is that it shifts the design complexity
 49 from the building material (the tiles) to the machinery (the environment). As recent practical work by
 50 Manzoor et al. [23] shows, it is possible to apply this to simple “sticky” particles that can be forced to bond,
 51 see Fig. 1: the overall assembly is achieved by adding particles one at a time, attaching them to the existing
 52 sub-assembly.

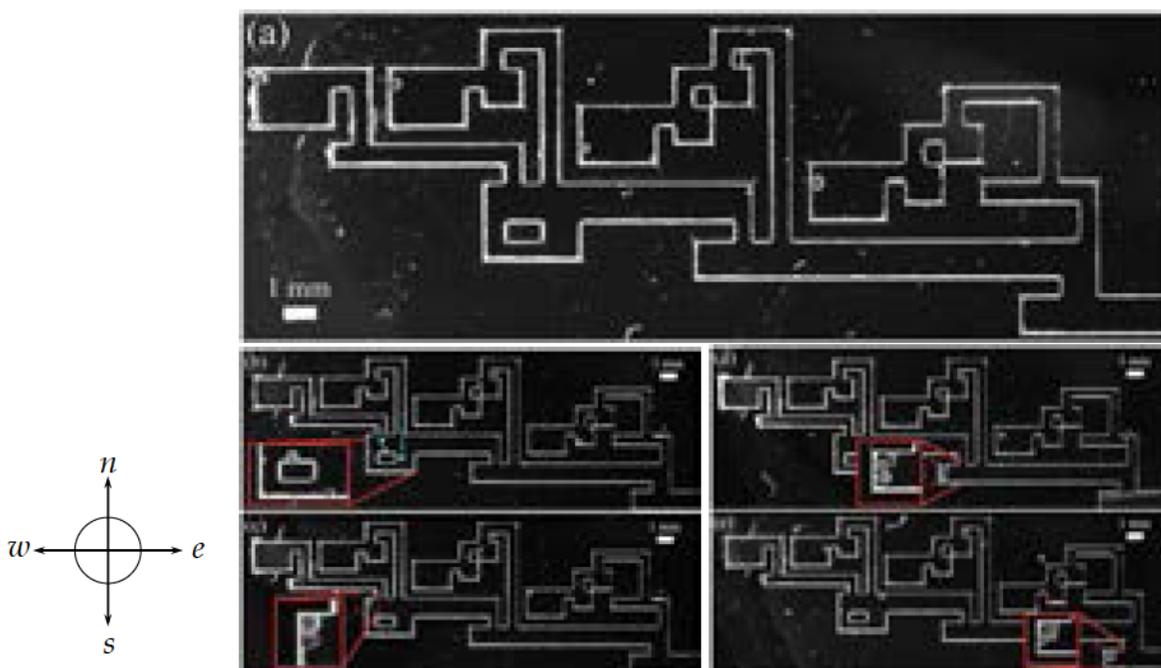


Figure 1: A practical demonstration of Tilt Assembly based on alginate (i.e., a gel made by combining a powder derived from seaweed with water) particles on a silicon wafer with obstacles etched in photoresist [23]. The obstacles appear as white lines and block the motion of particles. (a) Alginate particles in initial positions. (b) After control moves of $\langle e, s, w, n, e, s \rangle$ (for east, south, west, north), the alginate microrobots move to the shown positions. (c) After $\langle w, n \rangle$ inputs, the system produces the first multi-microrobot polyomino. (d) The next three microrobot polyominoes are produced after applying multiple $\langle e, s, w, n \rangle$ cycles. (e) After the alginate microrobots have moved through the microfluidic factory layout, the final 4-particle polyomino is generated.

53 Moreover, Manzoor et al. [23] argue that it is possible to enhance the overall assembly environment to
 54 allow pipelined construction, as shown in Fig. 2: after constructing the first polyomino, each cycle of a small
 55 control sequence produces another polyomino. However, the algorithmic part of [23] is purely heuristic;
 56 providing a thorough understanding of algorithms and complexity is the content of our paper.

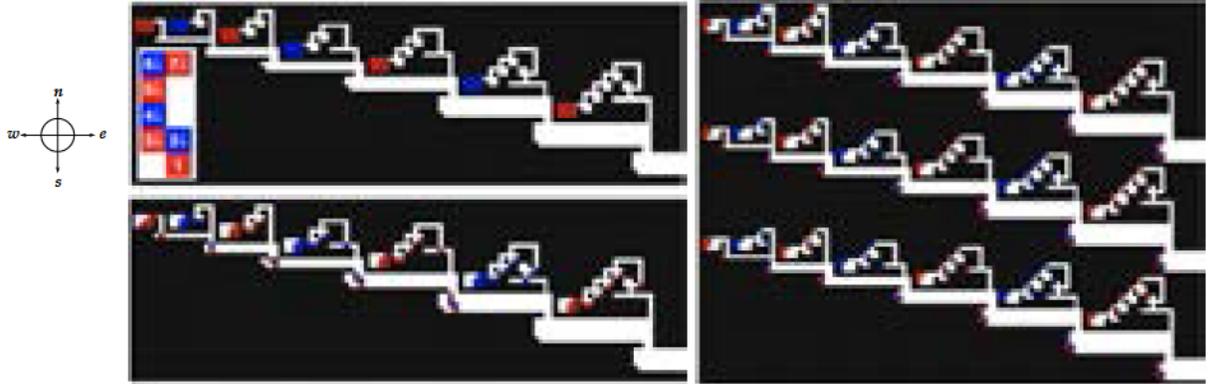


Figure 2: (Top left) Initial setup of a seven-tile polyomino assembly; the composed shape is shown enlarged on the lower left. The bipartite decomposition into blue and red particles is shown for greater clarity, but can also be used for better control of bonds. The sequence of control moves is $\langle e, s, w, n \rangle$, i.e., a clockwise order. (Bottom left) The situation after 18 control moves. (Right) The situation after 7 full cycles, i.e., after 28 control moves; shown are three parallel “factories”.

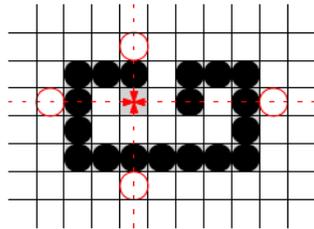


Figure 3: A polyomino (black) that cannot be constructed by Tilt Assembly: the last tile cannot be attached, as it gets blocked by previously attached tiles.

57 One critical issue of this approach is the requirement of getting particles to their destination without
 58 being blocked by or bonding to other particles. As Fig. 3 shows, this is not always possible, so there are some
 59 shapes that cannot be constructed by Tilt Assembly.

60 This gives rise to a variety of algorithmic questions: (1) Can we decide efficiently whether a given
 61 polyomino can be constructed by Tilt Assembly? (2) Can the resulting process be pipelined to yield low
 62 amortized building time? (3) Can we compute a maximum-size subpolyomino that can be constructed? (4)
 63 What can be said about three-dimensional versions of the problem?

64 1.1 Our Contribution

65 Our main contribution is a characterization of deciding constructibility and efficient construction for simply
 66 connected two-dimensional shapes: For a simple polyomino P with N pixels, we can decide in time $O(N \log N)$
 67 whether P can be constructed; using pipelining, a constructible polyomino can be built in amortized time
 68 $O(1)$. On the other hand, we show that deciding constructibility in 3D is NP-complete. We also provide
 69 a number of additional results on approximation and the constructibility of subpaths; see Table 1 for an
 70 overview.

71 1.2 Related Work

72 Assembling polyominoes with tiles has been considered intensively in the context of *tile self-assembly*. In
 73 1998, Erik Winfree [31] introduced the *abstract tile self-assembly model* (aTAM), in which tiles have glue

Dimension	Polyomino	Decision	Maximization	Approximation	Constructible Path
2D	simple	$O(N \log N)$ (Sec. 3)	<i>polyAPX</i> -hard	$O(N^{1/3}), \Omega(\sqrt{N})$ (Sec. 4)	$O(N \log N)$ (Sec. 4)
3D	general	<i>NP</i> -complete (Sec. 5)	<i>polyAPX</i> -hard	$O(N^{1/3}),$ - (Sec. 4)	<i>NP</i> -complete (Sec. 5)

Table 1: Results for Tilt Assembly Problem (TAP) and its maximization variant (MAXTAP)

types on each of the four sides and two tiles can stick together if their glue type matches and the bonding strength is sufficient. Starting with a *seed tile*, tiles will continue to attach to the existing partial assembly until they form a desired polyomino; the process stops when no further attachments are possible. For early examples of related work, see Rothmund and Winfree [24] and Adleman et al. [1] for the running time and program size for self-assembling squares. Apart from the aTAM, there are various other models like the *two-handed tile self-assembly model* (2HAM) [11] and the *hierarchical tile self-assembly model* [13], in which we have no single seed but pairs of subassemblies that can attach to each other. Furthermore, the *staged self-assembly model* [12, 14, 16] allows greater efficiency by assembling polyominoes in multiple bins which are gradually combined with the content of other bins.

All this differs from the model in Tilt Assembly, in which each tile has the same glue type on all four sides, and tiles are added to the assembly one at a time by attaching them from the outside along a straight line. This approach of externally movable tiles has been considered in practice at the microscale level using biological cells and an MRI, see [7, 20, 21]. Becker et al. [8] consider this for the assembly of a magnetic *Gauß gun*, which can be used for applying strong local forces by very weak triggers, allowing applications such as micro-surgery.

Using an external force for moving the robots becomes inevitable at some scale because the energy capacity decreases faster than the energy demand. A consequence is that all non-fixed robots/particles perform the same movement, so all particles move in the same direction of the external force until they hit an obstacle or another particle. These obstacles allow shaping the particle swarm. Designing appropriate sets of obstacles and moves gives rise to a range of algorithmic problems. Deciding whether a given initial configuration of particles in a given environment can be transformed into a desired target configuration is *NP*-hard [4], even in a grid-like setting, whereas finding an optimal control sequence is shown to be *PSPACE*-complete by Becker et al. [5]. However, if *designing* the obstacles is allowed, the problems become more tractable [4]. Moreover, even complex computations become possible: If we allow additional particles of double size (i.e., two adjacent fields), full computational complexity is achieved, see Shad et al. [26]. Further related work includes gathering a particle swarm at a single position [22] and using swarms of very simple robots (such as Kilobots) for moving objects [9]. For the case in which human controllers have to move objects by such a swarm, Becker et al. [6] study different control options. The results are used by Shahrokhi and Becker [27] to investigate an automatic controller.

The construction of polyominoes has also applications in the field of robot swarms, e.g., *shape formation*. Werfel and Nagpal [29, 30] show how multiple robots can move tiles to a partial assembly to construct a desired shape in 2D and 3D. Derakhshandeh et al. [17, 18] consider the robots as building material, which have $O(1)$ memory, and provide algorithms letting the robots form or coat shapes. In a very recent paper, Demaine et al [3, 15] show that a robot swarm can be reconfigured in time $O(d)$ unit steps, where d is the maximum distance of any robot. However, this requires the robots to be well separable. Arbutle and Requicha [2] show how a self-organized swarm of robots can construct a certain shape. In case of robot failures or external disturbance, the swarm is also able to repair the shape.

Further related work includes robots performing various tasks: Thubagere et al. [28] show that robots made from DNA can simultaneously sort molecular cargoes. Rubenstein et al. [25] consider a swarm of simple robots moving an object to a desired destination without knowing its shape and weight. Hoffmann [19] proves that it is *NP*-hard to decide if a robot can push its way through an area filled with blocks.

2 Preliminaries

Polyomino: For a set $P \subset \mathbb{Z}^2$ of N grid points in the plane, the graph G_P is the induced grid graph, in which two vertices $p_1, p_2 \in P$ are connected if they are at unit distance. Any set P with connected grid graph G_P gives rise to a *polyomino* by replacing each point $p \in P$ by a unit square centered at p , which is called a *tile*; for simplicity, we also use P to denote the polyomino when the context is clear, and refer to G_P as the dual graph of the polyomino; P is *tree-shaped* if G_P is a tree. A polyomino is called *hole-free* or *simple* if and only if the grid graph induced by $\mathbb{Z}^2 \setminus P$ is connected.

Blocking sets: For each point $p \in \mathbb{Z}^2$ we define *blocking sets* $N_p, S_p \subseteq P$ as the set of all points $q \in P$ that are above or below p and $|p_x - q_x| \leq 1$. Analogously, we define the blocking sets $E_p, W_p \subseteq P$ as the set of all points $q \in P$ that are to the right or to the left of p and $|p_y - q_y| \leq 1$.

Construction step: A *construction step* is defined by a direction $d \in \{\text{north, east, south, west}\}$ (abbreviated by n, e, s, w) from which a tile is added and a latitude/longitude l describing a column or row. The tile arrives from (l, ∞) for north, (∞, l) for east, $(l, -\infty)$ for south, and $(-\infty, l)$ for west into the corresponding direction until it reaches the first grid position that is adjacent to one occupied by an existing tile. If there is no such tile, the polyomino does not change. We note that a position p can be added to a polyomino P if and only if there is a point $q \in P$ with $\|p - q\|_1 = 1$ and one of the four blocking sets, N_p, E_p, S_p or W_p , is empty. Otherwise, if none of these sets are empty, this position is *blocked*.

Constructibility: Beginning with a seed tile at some position p , a polyomino P is *constructible* if and only if there is a sequence $\sigma = ((d_1, l_1), (d_2, l_2), \dots, (d_{N-1}, l_{N-1}))$, such that the resulting polyomino P' , induced by successively adding tiles with σ , is equal to P . We allow the constructed polyomino P' to be a translated copy of P . Reversing σ yields a *decomposition sequence*, i.e., a sequence of tiles removed from P .

3 Constructibility of Simple Polyominoes

In this section we focus on hole-free (i.e., simple) polyominoes. We show that the problem of deciding whether a given polyomino can be constructed can be solved in polynomial time. This decision problem can be defined as follows.

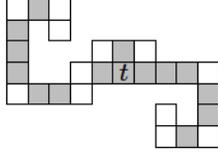
Definition 1 (TILT ASSEMBLY PROBLEM). *Given a polyomino P , the TILT ASSEMBLY PROBLEM (TAP) asks for a sequence of tiles constructing P , if P is constructible.*

3.1 A Key Lemma

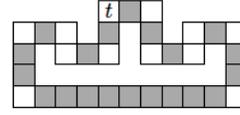
A simple observation is that construction and (connectivity-preserving) decomposition are the same problem. This allows us to give a more intuitive argument, as it is easier to argue that we do not lose connectivity when removing tiles than it is to prove that we do not block future tiles.

Theorem 2. *A polyomino P can be constructed if and only if it can be decomposed using a sequence of tile removal steps that preserve connectivity. A construction sequence is a reversed decomposition sequence.*

Proof. To prove this theorem, it suffices to consider a single step. Let P be a polyomino and t be a tile that is removed from P into some direction l , leaving a polyomino P' . Conversely, adding t to P' from direction l yields P , as there cannot be any tile that blocks t from reaching the correct position, or we would not be able to remove t from P in direction l . \square



(a) Removing t destroys decomposability. The polyomino can be decomposed by starting with the three tiles above t .



(b) Removing the locally convex tile t leaves the polyomino non-decomposable; it can be decomposed by starting from the bottom or the sides.

Figure 4: Two polyominoes and their locally convex tiles (white). (a) Removing not locally convex tiles may destroy decomposability. (b) With non-simple polygons we may not be able to remove locally convex tiles.

156 For hole-free polyominoes we can efficiently find a construction/decomposition sequence if one exists.
 157 The key insight is that one can greedily remove *locally convex* tiles. A tile t is said to be locally convex if
 158 and only if it is locally extremal, i.e., there are two axis-parallel orthogonal directions for which there is
 159 no tile connected to t ; see Fig. 4. If a locally convex tile is *not* a cut tile, i.e., it is a tile whose removal
 160 does *not* disconnect the polyomino, its removal does not interfere with the decomposability of the remaining
 161 polyomino.

162 This conclusion is based on the observation that a minimal cut (i.e., a minimal set of vertices whose
 163 removal leaves a disconnected polyomino) of cardinality two in a hole-free polyomino always consists of two
 164 (possibly diagonally) adjacent tiles. Furthermore, we can always find such a removable locally convex tile in
 165 any decomposable hole-free polyomino. This allows us to devise a simple greedy algorithm.

166 We start by showing that if we find a non-blocked locally convex tile that is not a cut tile, we can simply
 167 remove it. It is important to focus on locally convex tiles, as the removal of not locally convex tiles can harm
 168 the decomposability: see Fig. 4a for an illustration. In non-simple polyominoes, the removal of locally convex
 169 tiles can destroy decomposability, as demonstrated in Fig. 4b.

170 **Lemma 3.** Consider a non-blocked, non-cut, locally convex tile t in a hole-free polyomino P . The polyomino
 171 $P - t$ is decomposable if and only if P is decomposable.

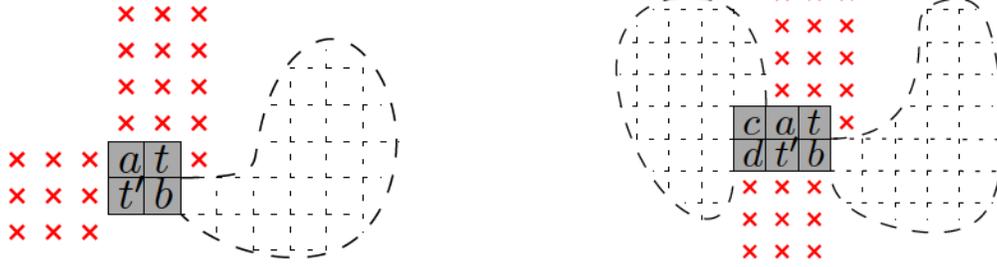
172 *Proof.* The first direction is trivial: if $P - t$ is decomposable, P is decomposable as well, because we can
 173 remove the non-blocked tile t first and afterwards use the existing decomposition sequence for $P - t$. The
 174 other direction requires some case distinctions. Suppose for contradiction that P is decomposable but $P - t$
 175 is not, i.e., t is important for the later decomposition.

176 Consider a valid decomposition sequence for P and the first tile t' we cannot remove if we were to remove
 177 t in the beginning. W.l.o.g., let t' be the first tile in this sequence (removing all previous tiles obviously does
 178 not destroy the decomposability). When we remove t first, we are missing a tile, hence t' cannot be blocked
 179 but has to be a cut tile in the remaining polyomino $P - t$. The presence of t preserves connectivity, i.e.,
 180 $\{t, t'\}$ is a minimal cut on P . Because P has no holes, then t and t' must be diagonal neighbors, sharing the
 181 neighbors a and b . Furthermore, by definition neither of t and t' is blocked in some direction. We make a
 182 case distinction on the relation of these two directions.

183 **The directions are orthogonal (Fig. 5a).** Either a or b is a non-blocked locally convex tile, because t
 184 and t' are both non-blocked; w.l.o.g., let this be a . It is easy to see that independent of removing t or t'
 185 first, after removing a we can also remove the other one.

186 **The directions are parallel (Fig. 5b).** This case is slightly more involved. By assumption, we have a
 187 decomposition sequence beginning with t' . We show that swapping t' with our locally convex tile t in
 188 this sequence preserves feasibility.

189 The original sequence has to remove either a or b before it removes t , as otherwise the connection
 190 between the two is lost when t' is removed first. After either a or b is removed, t becomes a leaf and
 191 can no longer be important for connectivity. Thus, we only need to consider the sequence until either a



(a) If the unblocked directions of t and t' are orthogonal, one of the two adjacent tiles (w.l.o.g. a) cannot have any further neighbors. There can also be no tiles in the upper left corner, because the polyomino cannot cross the two free directions of t and t' (red marks).

(b) If the unblocked directions of t and t' are parallel, there is only the tile c for which something can change if we remove t before t' .

Figure 5: The red marks indicate that no tile is at this position; the dashed outline represents the rest of the polyomino.

192 or b is removed. The main observation is that a and b block the same tiles as t or t' , except for tile c as
 193 in Fig. 5b. However, when c is removed, c has to be a leaf, because a is still not removed and in the
 194 original decomposition sequence, t' has already been removed. Therefore, a tile $d \neq t'$ would have to be
 195 removed before c . Hence, the decomposition sequence remains feasible, concluding the proof. \square

Next we show that such a locally convex tile always exists if the polyomino is decomposable.



(a) If the removal direction of t is not crossed, the last blocking tile has to be locally convex (and has to be removed before t).

(b) If the removal direction of t crosses P , then P gets split into components A and B . Component B has a locally convex tile t' that needs to be removed before t .

Figure 6: Polyominoes for which no locally convex tile should be removable, showing the contradiction to t being the first blocked locally convex tile in P removed.

196

197 **Lemma 4.** *Let P be a decomposable polyomino. Then there exists a locally convex tile that is removable*
 198 *without destroying connectivity.*

199 *Proof.* We prove this by contradiction based on two possible cases.

200 Assume P to be a decomposable polyomino in which no locally convex tile is removable. Because P is
 201 decomposable, there exists some feasible decomposition sequence S . Let P_{convex} denote the set of locally
 202 convex tiles of P and let $t \in P_{\text{convex}}$ be the first removed locally convex tile in the decomposition sequence S .
 203 By assumption, t cannot be removed yet, so it is either blocked or a cut tile.

204 t is blocked. Consider the direction in which we would remove t . If it does not cut the polyomino, the last
 205 blocking tile has to be locally convex (and would have to be removed before t), see Fig. 6a. If it cuts
 206 the polyomino, the component cut off also must have a locally convex tile and the full component has
 207 to be removed before t , see Fig. 6b. This is again a contradiction to t being the first locally convex tile
 208 to be removed in S .

209 t is a cut tile. $P - t$ consists of exactly two connected polyominoes, P_1 and P_2 . It is easy to see that
 210 $P_1 \cap P_{\text{convex}} \neq \emptyset$ and $P_2 \cap P_{\text{convex}} \neq \emptyset$, because every polyomino of size $n \geq 2$ has at least two locally
 211 convex tiles of which at most one ceases to be locally convex by adding t . (A polyomino of size 1 is
 212 trivial.) Before being able to remove t , either P_1 or P_2 has to be completely removed, including their
 213 locally convex tiles. This is a contradiction to t being the first locally convex tile in S to be removed. \square

214 3.2 An Efficient Algorithm

215 An iterative combination of these two lemmas proves the correctness of greedily removing locally convex tiles.
 216 As we show in the next theorem, using a search tree technique allows an efficient implementation of this
 217 greedy algorithm.

218 **Theorem 5.** *A hole-free polyomino can be checked for decomposability/constructibility in time $O(N \log N)$.*

219 *Proof.* Lemma 3 allows us to remove any locally convex tile, as long as it is not blocked and does not
 220 destroy connectivity. Applying the same lemma on the remaining polyomino iteratively creates a feasible
 221 decomposition sequence. Lemma 4 proves that this is always sufficient. If and only if we can at some point
 222 no longer find a matching locally convex tile (to which we refer as *candidates*), the polyomino cannot be
 223 decomposable.

224 Let B be the time needed to check whether a tile t is blocked. A naïve way of doing this is to try out
 225 all tiles and check if t gets blocked, requiring time $O(N)$. With a preprocessing step, we can decrease B to
 226 $O(\log N)$ by using $O(N)$ binary search trees for searching for blocking tiles and utilizing that removing a tile
 227 can change the state of at most $O(1)$ tiles. For every vertical line x and horizontal line y going through P ,
 228 we create a balanced search tree, i.e., for a total of $O(N)$ search trees. An x -search tree for a vertical line x
 229 contains tiles lying on x , sorted by their y -coordinate. Analogously define a y -search tree for a horizontal line
 230 y containing tiles lying on y sorted by their x -coordinate. We iterate over all tiles $t = (x, y)$ and insert the
 231 tile in the corresponding x - and y -search tree with a total complexity of $O(N \log N)$. Note that the memory
 232 complexity remains linear, because every tile is in exactly two search trees. To check if a tile at position
 233 (x', y') is blocked from above, we can simply search in the $(x' - 1)$ -, x' - and $(x' + 1)$ -search tree for a tile with
 234 $y > y'$. We analogously perform search queries for the other three directions, and thus have 12 queries of
 235 total cost $O(\log N)$.

236 We now iterate on all tiles and add all locally convex tiles that are not blocked and are not a cut tile to
 237 the set F (cost $O(N \log N)$). Note that checking whether a tile is a cut tile can be done in constant time,
 238 because it suffices to look into the local neighborhood. While F is not empty, we remove a tile from F , from
 239 the polyomino, and from its two search trees in time $O(\log N)$. Next, we check the up to 12 tiles that could
 240 have been blocked by the removed tile, see Fig. 7. Only these tiles can become unblocked or a locally convex
 241 tile. Those that are locally convex tiles, not blocked, and not a cut tile are added to F . All tiles behind
 242 those cannot become unblocked as the first tiles would still be blocking them. If one of those tiles becomes
 243 a cut tile, then we remove it from F . The cost for this is again in $O(\log N)$. This is continued until F is
 244 empty, which takes at most $O(N)$ loops each of cost $O(\log N)$. If the polyomino has been decomposed, the
 245 polyomino is decomposable/constructible by the corresponding tile sequence. Otherwise, there cannot exist
 246 such a sequence. A specific start tile can be enforced by prohibiting the removal of that tile. \square

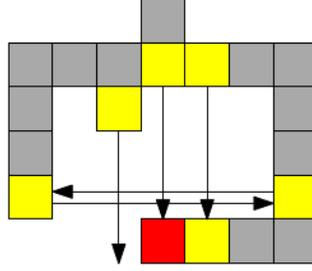


Figure 7: When removing the red tile (dark gray in grayscale), only the yellow tiles (light gray in grayscale) can become unblocked or locally convex.

3.3 Pipelined Assembly

Given that a construction is always possible based on adding locally convex corners to a partial construction, we can argue that the heuristic idea of Manzoor et al. [23] for pipelined assembly can be formally realized for every constructible polyomino: We can transform the construction sequence into a spiral-shaped maze environment, as illustrated in Fig. 8. This allows it to produce D copies of P in $N + D$ cycles, implying that we only need $2N$ cycles for N copies. It suffices to use a clockwise order of four unit steps (west, north, east, south) in each cycle.

The main idea is to create a spiral in which the assemblies move from the inside to the outside. The first tile is provided by an initial south movement. After each cycle, ending with a south movement, the next seed tile of the next copy of P is added. For every direction corresponding to the direction of the next tile added by the sequence, we place a tile depot on the outside of the spiral, with a straight-line path to the location of the corresponding attachment.

Theorem 6. *Given a construction sequence $\sigma := ((d_1, l_1), \dots, (d_{N-1}, l_{N-1}))$ that constructs a polyomino P , we can construct a maze environment for pipelined tilt assembly, such that constructing D copies of P needs $O(N + D)$ unit steps. In particular, constructing one copy of P can be done in amortized time $O(1)$.*

Proof. Consider the construction sequence σ , the movement sequence ζ consisting of N repetitions of the cycle (w, n, e, s) , and an injective function $m : \sigma \rightarrow \zeta$, with $m((w, \cdot)) = e$, $m((n, \cdot)) = s$, $m((e, \cdot)) = w$ and $m((s, \cdot)) = n$. We also require that $m((d_i, l_i)) = \zeta_j$ if for all $i' < i$ there is a $j' < j$ with $m((d_{i'}, l_{i'})) = \zeta_{j'}$ and j is the smallest possible. This implies that in each cycle there is at least one tile in σ mapped to one direction in this cycle.

Labyrinth construction: The main part of the labyrinth is a spiral as can be seen in Fig. 8. Consider a spiral that is making $|\zeta|$ many turns, and the innermost point q of this spiral. From q upwards, we make a lane through the spiral until we are outside the spiral. At this point we add a depot of tiles, such that after each south movement a new tile comes out of the depot (this can easily be done with bottleneck constructions as seen in Fig. 8). Then, we proceed for each turn in the spiral as follows: For the j -th turn, if $m^{-1}(\zeta_j)$ is empty we do nothing. Else if $m^{-1}(\zeta_j)$ is not empty we want to add the next tile. Let t_i be this particular tile. Then, we construct a lane in direction $-\zeta_j$, i.e., the direction from where the tile will come from, until we are outside the spiral. By shifting this line in an orthogonal direction we can enforce the tile to fly in at the correct position relating to l_i . There, we add a depot with tiles, such that the first tile comes out after $j - 1$ steps and with each further cycle a new tile comes out (this can be done by using loops in the depot, see Fig. 8). Depots, which lie on the same side of the spiral, can be shifted arbitrarily, so they do not collide. These depots can be made arbitrarily big, and thus, we can make as many copies of P as we wish. Note that we can make the paths in the spiral big enough, such that after every turn the bounding box of the current polyomino fits through the spiral.

Correctness: We will now show that we will obtain copies of P . Consider any j -th turn in the spiral, where the i -th tile t_i is going to be added to the current polyomino. With the next step, both t_i and

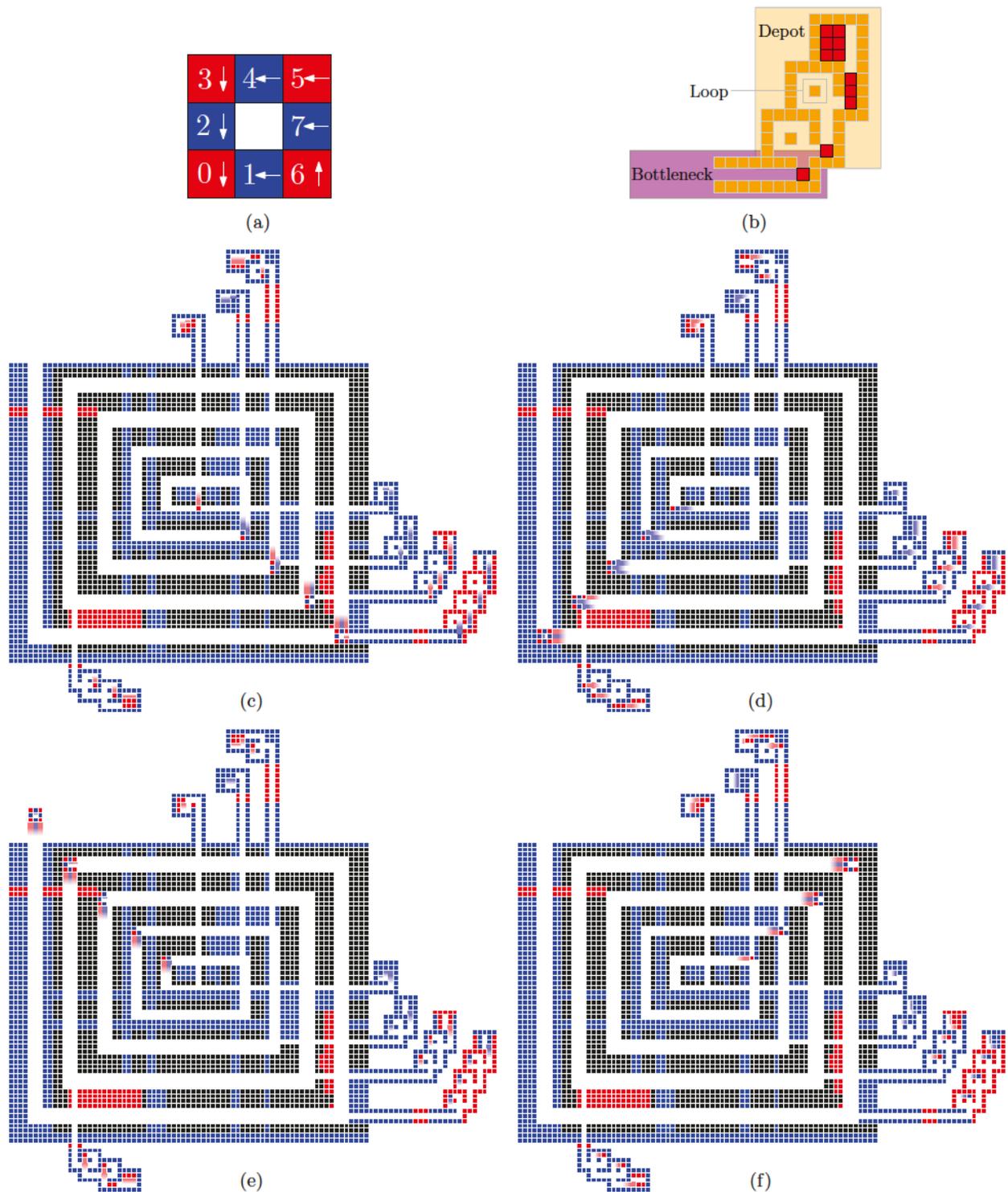


Figure 8: (a) A polyomino P . Shown is the assembly order and the direction of attachment to the seed (tile 0). (b) A depot (orange; light gray area in grayscale) having loops to delay the tile output and a bottleneck (purple; dark gray area in grayscale) to guarantee that only one tile can move to the spiral. (c to f) A maze environment for pipelined construction of the desired polyomino P . After the fourth cycle, each further cycle produces a new copy of P . Shown states are after a sequence of down (c), left (d), up (e) and right (f) moves.



Figure 9: Two different sequences. The red tile represents the bounding box of the current polyomino. (Left) A desired sequence. The latitude intersects the bounding box. (Right) A sequence where the latitude does not intersect the bounding box.

283 the polyomino move in direction ζ_j . While the polyomino does not touch the next wall in the spiral,
 284 the distance between t_i and the polyomino will not decrease. However when the polyomino hits the
 285 wall, the polyomino stops moving and t_i continues moving towards the polyomino. Wall-hitting is the
 286 same situation as in our non-parallel model: To a fixed polyomino we can add tiles from n , e , s or
 287 w . Therefore, the tile connects to the correct place. Since this is true for any tile and any copy, we
 288 conclude that every polyomino we build is a copy of P .

289 **Time:** Since the spiral has at most $4N$ unit steps (or N cycles), the first polyomino will be constructed
 290 after $4N$ unit steps. By construction, we began the second copy one cycle after beginning the first
 291 copy, the third copy one cycle after the second, and so on. This means, after each cycle, when the first
 292 polyomino is constructed, we obtain another copy of P . Therefore, for D copies we need $N + D$ cycles
 293 (or $O(N + D)$ unit steps). For $D \in \Omega(N)$ this results in an amortized constant time construction for P .

294 Note that this proof only considers construction sequences in the following form: If a tile t_i increases the
 295 side length of the bounding box of the current polyomino, then the tile is added from a direction with a
 296 longitude/latitude, such that the longitude/latitude intersects the bounding box (see Fig. 9). In the case
 297 there is a tile, such that the longitude/latitude does not intersect the bounding box, then we can rotate the
 298 direction by $\frac{\pi}{2}$ towards the polyomino and we will have a desired construction sequence. \square

299 4 Optimization Variants in 2D

300 For polyominoes that cannot be assembled, it is natural to look for a maximum-size subpolyomino that
 301 is constructible. This optimization variant is *polyAPX-hard*, i.e., we cannot hope for an approximation
 302 algorithm with an approximation factor within $\Omega(N^{\frac{1}{3}})$, unless $P = NP$.

303 **Definition 7** (Maximum Tilt Assembly Problem). *Given a polyomino P , the Maximum Tilt Assembly*
 304 *Problem (MAXTAP) asks for a sequence of tiles building a cardinality-maximal connected subpolyomino*
 305 *$P' \subseteq P$.*

306 **Theorem 8.** *MAXTAP is polyAPX-hard, even for tree-shaped polyominoes.*

307 *Proof.* We reduce MAXIMUM INDEPENDENT SET (MIS) to MAXTAP; see Fig. 10 for an illustration. Consider
 308 an instance $G = (V, E)$ of MIS, which we transform into a polyomino P_G . We construct P_G as follows. First,
 309 construct a horizontal line from which we go down to select which vertex in G will be chosen. The line must
 310 have length $10n - 9$, where $n = |V|$. Every 10th tile will represent a vertex, starting with the first tile on the
 311 line. Let t_i be such a tile representing vertex v_i . For every v_i we add a selector gadget below t_i and for every
 312 $\{v_i, v_j\} \in \delta(v_i)$ we add a reflected selector gadget below t_j , as shown in Fig. 10, each consisting of 19 tiles.
 313 Note that all gadgets for selecting vertex v_i are above the gadgets of v_j if $i < j$ and that there are at most n^2
 314 such gadgets. After all gadgets have been constructed, we have already placed at most $19n^2 + 10n - 9 \leq 29n^2$
 315 tiles. We continue with a vertical line with a length of $30n^2$ tiles.

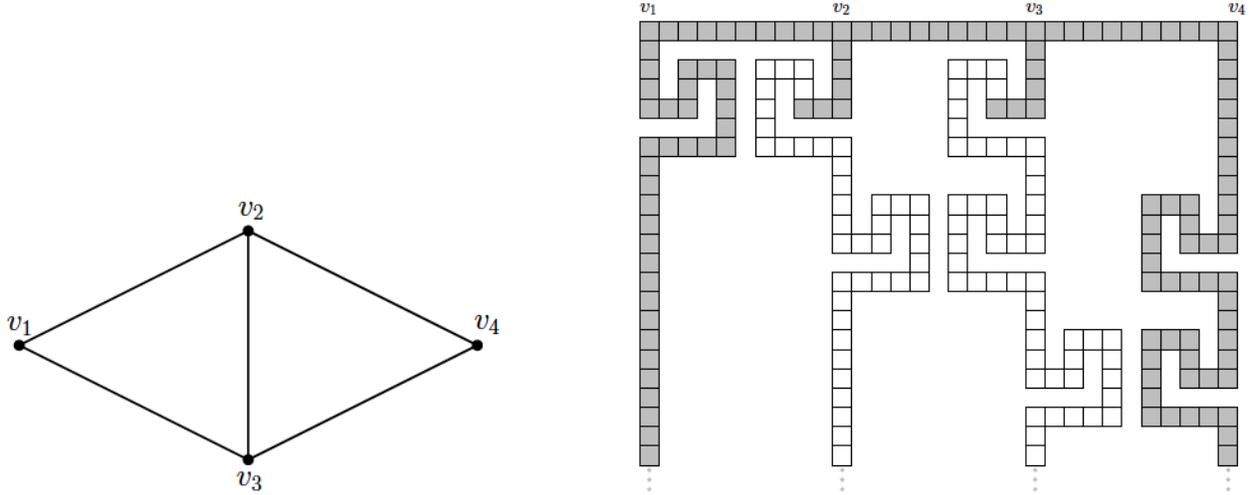


Figure 10: Reduction from MIS to MAXTAP. (Left) A graph G with four vertices. (Right) A polyomino constructed for the reduction with a feasible, maximum solution marked in gray.

316 Now, let α^* be an optimal solution to MIS. Then MAXTAP has a maximum polyomino of size at least
 317 $30n^2\alpha^*$ and at most $30n^2\alpha^* + 29n^2$: We take the complete vertical part of t_i for every v_i in the optimal
 318 solution of MIS. Choosing other lines block the assembly of further lines and thus, yields a smaller solution.

319 Now suppose we had an $N^{1-\varepsilon}$ -approximation for MAXTAP. Then we would have a solution of at least
 320 $\frac{1}{N^{1-\varepsilon}}T^*$, where T^* is the optimal solution. We know that an optimal solution has $T^* \geq 30n^2\alpha^*$ tiles and the
 321 polyomino has at most $N \leq 30n^3 + 29n^2 \leq 59n^3$ tiles. Therefore, we have at least $\frac{30n^2\alpha^*}{59^{1-\varepsilon}n^{3-3\varepsilon}}$ tiles and thus
 322 at least $\frac{1}{59^{1-\varepsilon}n^{3-3\varepsilon}}\alpha^*$ strips, because each strips is $30n^2$ tiles long. Consider some $\varepsilon \geq \frac{2}{3} + \eta$ for any $\eta > 0$,
 323 then the number of strips is $\frac{1}{59^{1/3}n^{1-3\eta}}\alpha^*$ which results in an $n^{1-\delta}$ -approximation for MIS, contradicting the
 324 inapproximability of MIS (unless $P=NP$) shown by Berman and Schnitger [10]. \square

325 As a consequence of the construction, we get Corollary 9.

326 **Corollary 9.** *Unless $P = NP$, MAXTAP cannot be approximated within a factor of $\Omega(N^{\frac{1}{3}})$.*

327 On the positive side, we can give an $O(\sqrt{N})$ -approximation algorithm for tree-shaped polyominoes.

328 **Theorem 10.** *The longest constructible path in a tree-shaped polyomino P is a \sqrt{N} -approximation for
 329 MAXTAP, and we can find such a path in polynomial time.*

330 *Proof.* Consider an optimal solution P^* and a smallest enclosing box B containing P^* . Then there must be
 331 two opposite sides of B touching at least one tile of P^* . Consider the path S between both tiles. Because
 332 (i) the area A_B of B is at least the number of tiles in P^* , (ii) $|S| \geq \sqrt{A_B}$, and (iii) a longest, constructible
 333 path in P has length at least $|S|$, we conclude that the longest constructible path is a \sqrt{N} -approximation.

334 To find such a path, we can search for every path between two tiles, check whether we can build this path,
 335 and take the longest, constructible path. \square

336 Checking constructibility for $O(N^2)$ possible paths is rather expensive. However, we can efficiently approx-
 337 imate the longest constructible path in a tree-shaped polyomino with the help of *sequentially* constructible
 338 paths, i.e., the initial tile is a leaf in the final path.

339 **Theorem 11.** *We can find a constructible path in a tree-shaped polyomino in $O(N^2 \log N)$ time that has a
 340 length of at least half the length of the longest constructible path.*

341 *Proof.* We only search for paths that can be built sequentially. Clearly, the longest such path is at least half
 342 as long as the longest path that can have its initial tile anywhere. We use the same search tree technique as
 343 before to look for blocking tiles. Select a tile of the polyomino as the initial tile. Do a depth-first search and
 344 for every tile in this search, check if it can be added to the path. If it cannot be added, skip all deeper tiles,
 345 as they also cannot be added. During every step in the depth-first search, we only need to change a single
 346 tile in the search trees, doing $O(1)$ updates with $O(\log N)$ cost. As we only consider $O(N)$ vertices in the
 347 depth-first search, this results in a cost of $O(N \log N)$ for a fixed start tile. It is trivial to keep track of the
 348 longest such constructible path. Repeating this for every tile results in a running time of $O(N^2 \log N)$. \square

349 In tree-shaped polyominoes, finding a constructible path is easy. For simple polyominoes, additional
 350 arguments and data structures lead to a similar result.

351 **Theorem 12.** *In simple polyominoes, finding the longest of all shortest paths that are sequentially constructible*
 352 *takes $O(N^2 \log N)$ time.*

353 Before we start with the proof of Theorem 12, we show in the next two lemmas that it is sufficient to
 354 consider shortest paths only, and that we can restrict ourselves to one specific shortest path between two
 355 tiles. Hence, we just need to test a maximum of $O(n^2)$ different paths.

356 **Lemma 13.** *In a sequentially constructible path, if there is a direct straight connection for a subpath, the*
 357 *subpath can be replaced by the straight connection.*

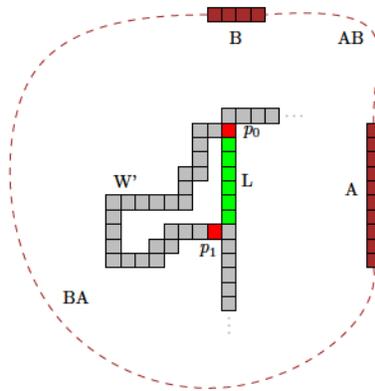


Figure 11: A subpath W' and its shortcut L in green. To block L , A and B must exist. But then, either p_0 or p_1 (red tiles) will also be blocked. Therefore, also W' cannot be built.

358 *Proof.* Consider a sequentially constructible path W and a subpath $W' \subset W$ that has a straight line L
 359 connecting the startpoint and the endpoint of W' . W.l.o.g., L is a vertical line and we build from bottom to
 360 top. Assume that $(W \setminus W') \cup L$ is not constructible. Then at least two structures (which can be single tiles)
 361 A and B must exist, preventing us from building L . Furthermore, these structures have to be connected via
 362 a path (AB or BA , see Fig. 11). We observe that none of these connections can exist or otherwise, we cannot
 363 build W (if AB exist, we cannot build the last tile p_0 of L ; if BA exist, we cannot build the first tile p_1 of
 364 W'). Therefore, we can replace W' with L . \square

365 By repeating the construction of Lemma 13 we get a shortest path from tile t_1 to t_2 in the following
 366 form: Let P_1, \dots, P_k be reflex tiles on the path from t_1 to t_2 . Furthermore, for every $1 \leq i \leq k - 1$, the path
 367 from P_i to P_{i+1} is monotone. This property holds for every shortest path, or else we can use shortcuts as in
 368 Lemma 13.

369 **Lemma 14.** *If a shortest path between two tiles is sequentially constructible, then every shortest path between*
 370 *these two tiles is sequentially constructible.*

371 *Proof.* Consider a constructible shortest path W , a maximal subpath W' that is x - y -monotone, and a
 372 bounding box B around W' . Due to the L_1 -metric, any x - y -monotone path within B is as long as W' .
 373 Suppose some path within B is not constructible. Then we can use the same blocking argument as in
 374 Lemma 13 to prove that W' cannot be constructible as well, contradicting that W is constructible. \square

375 Using Lemma 13 and Lemma 14, we are ready to prove Theorem 12.

376 **Proof of Theorem 12.** Because it suffices to check one shortest path between two tiles, we can look at
 377 the BFS tree from each tile and then proceed like we did in Theorem 11. Thus, for each tile we perform a
 378 BFS in time $O(N)$ and a DFS with blocking look-ups in time $O(N \log N)$, which results in a total time of
 379 $O(N^2 \log N)$. \square

380 5 Three-Dimensional Shapes

381 An interesting and natural generalization of TAP is to consider three-dimensional shapes, i.e., polycubes. The
 382 local considerations for simply connected two-dimensional shapes are no longer sufficient. In the following
 383 we show that deciding whether a polycube is constructible is NP -hard. Moreover, it is NP -hard to check
 384 whether there is a constructible path from a start cube s to an end cube t in a partial shape.

385 As a stepping stone, we start with a restricted version of the three-dimensional problem.

386 **Theorem 15.** *It is NP -hard to decide if a polycube can be built by inserting tiles only from above, north,*
 387 *east, south, and west.*

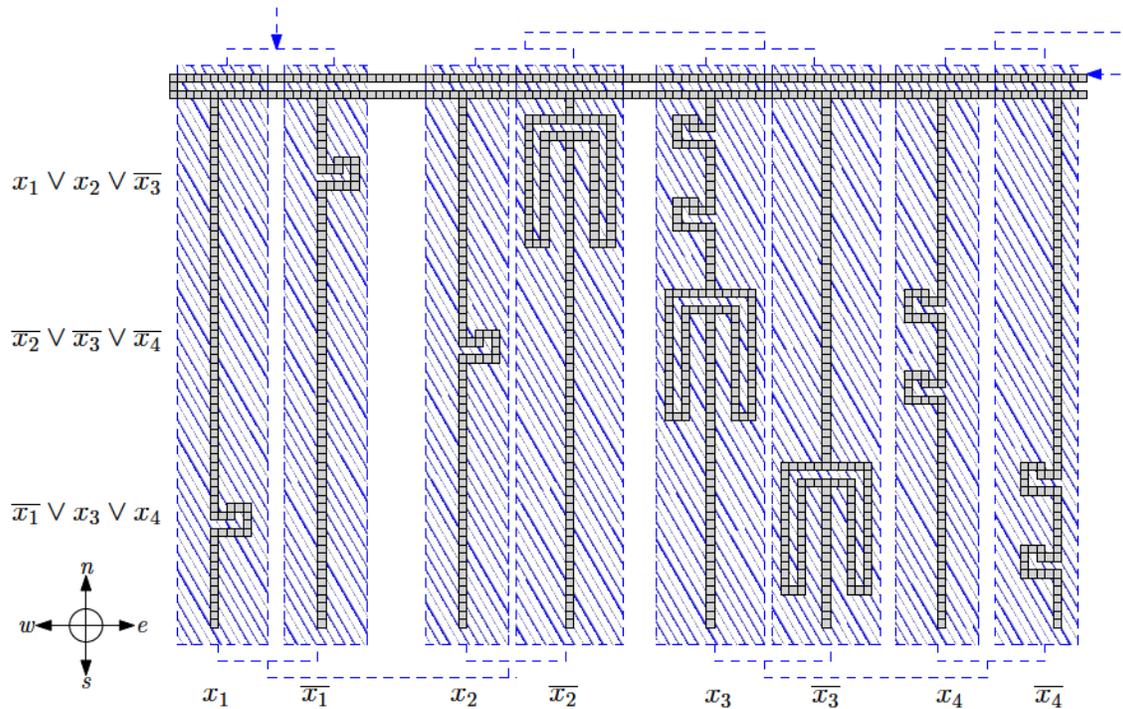


Figure 12: Top-view on the polycube. There is a vertical part going south for the *true* and *false* assignment of each variable. We start building at the top layer (crosshatched area) and have to block either the *true* or the *false* part of each variable from above. The blocked parts have to be built with only inserting from east, west, and south. For each clause, the parts of the inverted literals are modified to allow at most two of them being built in this way. All other parts can simply be inserted from above in the end.

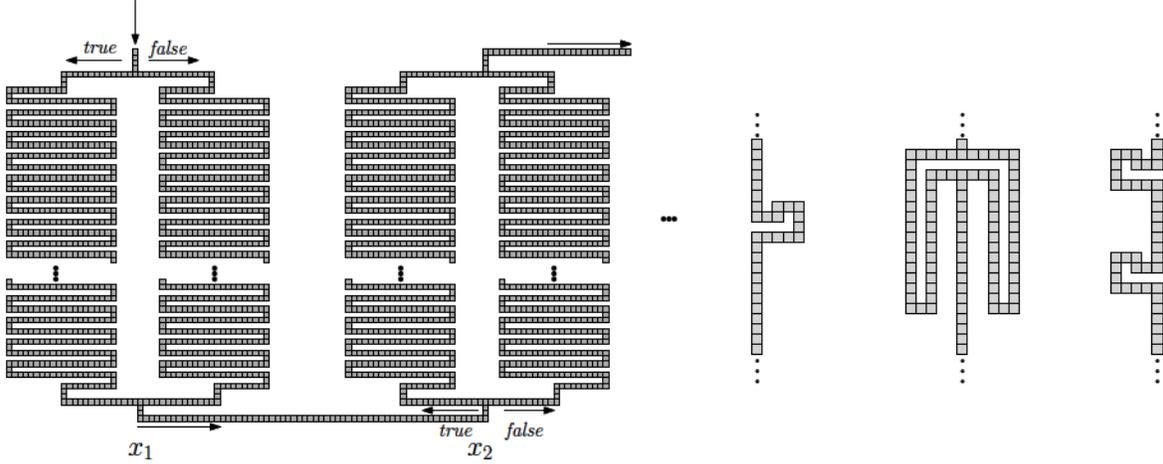


Figure 13: Top-view on the polycube. (Left) In the beginning we have to block the access from the top for either the *true* or *false* part of the variable. The variable is assigned the blocked value. (Right) Three gadgets for a clause. Only two of them can be built if the tiles are only able to come from the east, south, and west.

388 *Proof.* We prove hardness by a reduction from 3SAT. A visualization for the formula $(x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_2 \vee$
 389 $\bar{x}_3 \vee \bar{x}_4) \wedge (\bar{x}_1 \vee x_3 \vee x_4)$ can be seen in Fig. 12. It consists of two layers of interest (and some further auxiliary
 390 ones for space and forcing the seed tile by using the one-way gadget shown in Fig. 14). Due to the one-way
 391 gadget, at least part of the top layer (crosshatched area in Fig. 12, details in Fig. 13 (Left)) must be built
 392 first. Forcing a specific start tile can be done by a simple construction. For each variable we have to choose
 393 to block the left (for assigning *true*) or the right (for assigning *false*) part of the lower layer. In the end, the
 394 remaining parts of the upper layer can trivially be filled from above. The blocked parts of the lower layer
 395 then have to be built with only inserting tiles from east, south, or west. In the end, the non-blocked parts
 396 can be filled in from above. For each clause we use a part (as shown in Fig. 13 (Right)) that allows only at
 397 most two of its three subparts to be built from the limited insertion directions. We attach these subparts to
 398 the three variable values not satisfying the clause, i.e., the negated literals. This forces us to leave at least
 399 one negated literal of the clause unblocked, and thus at least one literal of the clause to be *true*. Overall, this
 400 allows us to build the blocked parts of the lower layers only if the blocking of the upper level corresponds to
 401 a satisfying assignment. If we can build the *true* and the *false* parts of a variable in the beginning, any truth
 402 assignment for the variable is possible.

403 It is straightforward to see that the whole construction fits into a bounding box of size $O(|C|) \times O(|V|) \times O(1)$,
 404 where C is the set of all clauses and V the set of all variables. \square

405 The construction can be extended to assemblies with arbitrary direction.

406 **Theorem 16.** *It is NP-complete to decide if a polycube can be built by inserting tiles from any direction.*

407 *Proof.* We add an additional layer below the construction in Theorem 15 that has to be built first and blocks
 408 access from below. Forcing the bottom layer to be built first can again be done with the one-way gadget
 409 shown in Fig. 14. Finally, we note that the problem of deciding whether a polycube can be built by inserting
 410 tiles from any direction is in NP. \square

411 The difficulties of construction in 3D are highlighted by the fact that even identifying constructible
 412 connections between specific positions is NP-hard.

413 **Theorem 17.** *It is NP-complete to decide whether a path from one tile to another can be built in a general*
 414 *polycube.*

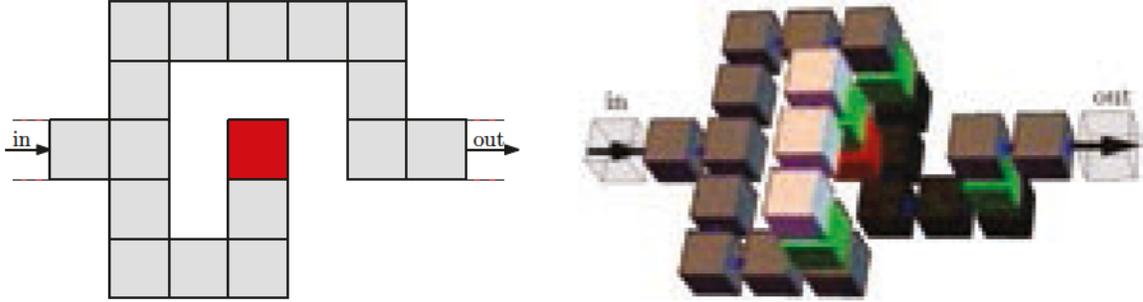


Figure 14: (Left) This polyomino can only be constructed by starting at “in” and ending at “out”. (Right) Generalization to three dimensions. If we start on the right side, then we cannot build the red cube because it is blocked from all six directions. With these gadgets we can enforce a seed tile.

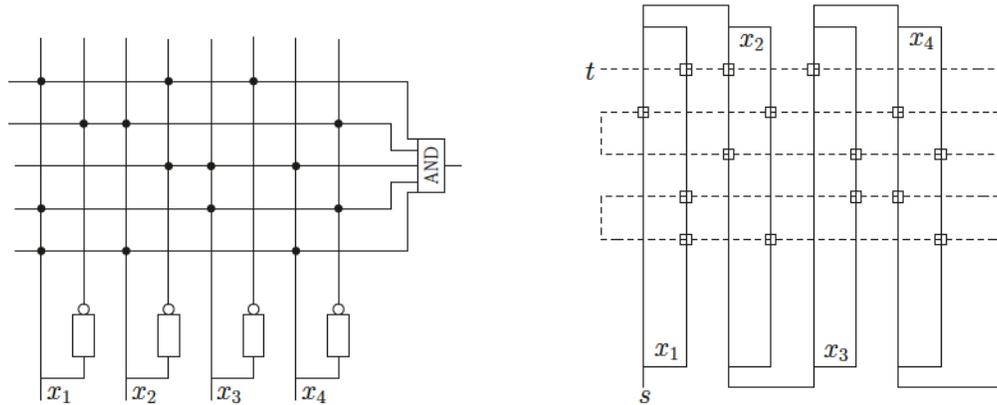


Figure 15: (Left) Circuit representation for the SAT formula $(x_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee x_2 \vee \bar{x}_4) \wedge (\bar{x}_2 \vee x_3 \vee x_4) \wedge (x_1 \vee x_3 \vee \bar{x}_4) \wedge (x_1 \vee x_2 \vee x_4)$. (Right) Reduction from SAT formula. Boxes represent variable boxes.

415 *Proof.* We prove *NP*-hardness by a reduction from SAT. For each variable we have two vertical lines, one
 416 for the *true* setting, one for the *false* setting. Each clause gets a horizontal line and is connected with a
 417 variable if it appears as literal in the clause, see Fig 15 (Left). We transform this representation into a tour
 418 problem where, starting at a point s , one first has to go through either the *true* or *false* line of each variable
 419 and then through all clause lines, see Fig. 15 (Right). The clause part is only passable if the path in at least
 420 one crossing part (squares) does not cross, forcing us to satisfy at least one literal of a clause. As one has to
 421 go through all clauses, t is only reachable if the selected branches for the variables equal a satisfying variable
 422 assignment for the formula.

423 We now consider how to implement this as a polycube. The only difficult part is to allow a constructible
 424 clause path if there is a free crossing. In Fig. 16 (Left), we see a variable box that corresponds to the crossing
 425 of the variable path at the squares in Fig. 15 (Right). It blocks the core from further insertions. The clause
 426 path has to pass at least one of these variable boxes in order to reach the other side. See Fig. 15 (Right) for
 427 an example. Note that the corresponding clause parts can be built by inserting only from above and below,
 428 so there are no interferences. \square

429 6 Conclusion/Future Work

430 We have provided a number of algorithmic results for Tilt Assembly. Various unsolved challenges remain.
 431 What is the complexity of deciding TAP for non-simple polyominoes? While Lemma 4 can be applied to all

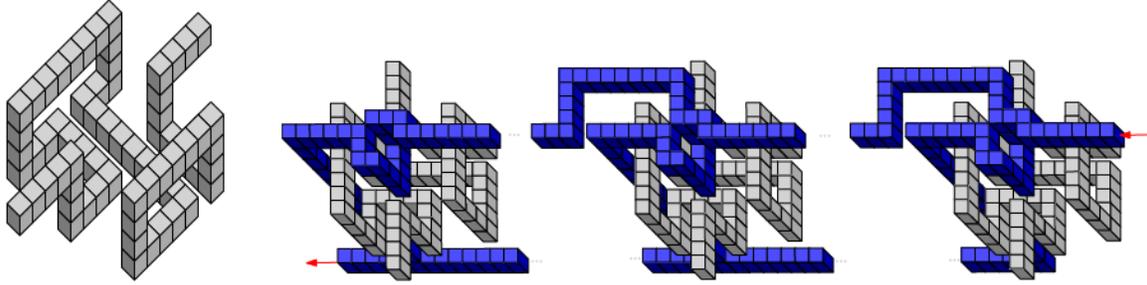


Figure 16: (Left) Empty variable box. (Right) A clause line (blue, dark gray in grayscale) dips into a variable box. If the variable box is built, then we cannot build the dip of the clause line.

432 polyominoes, we cannot simply remove any locally convex tile. Can we find a constructible path in a general
 433 polyomino from a given start and endpoint? This would help in finding a \sqrt{N} -approximation for non-simple
 434 polyominoes. How can we optimize the total makespan for constructing a shape? And what options exist for
 435 non-constructible shapes?

436 An interesting approach may be to consider *staged* assembly, as shown in Fig. 17, where a shape gets
 437 constructed by putting together subpolyominoes, instead of adding one tile at a time. This is similar to staged
 438 tile self-assembly [12, 14, 16]. This may also provide a path to sublinear assembly times, as a hierarchical
 439 assembly allows massive parallelization. We conjecture that a makespan of $O(\sqrt{N})$ for a polyomino with N
 440 tiles can be achieved.

441 All this is left to future work.



Figure 17: (Left) A polyomino that cannot be constructed in the basic TAP model. (Right) Construction in a staged assembly model by putting together subpolyominoes.

442 References

- 443 [1] L. Adleman, Q. Cheng, A. Goel, and M.-D. Huang. Running time and program size for self-assembled
 444 squares. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 740–748, 2001.
- 445 [2] D. Arbuckle and A. A. Requicha. Self-assembly and self-repair of arbitrary shapes by a swarm of reactive
 446 robots: algorithms and simulations. *Autonomous Robots*, 28(2):197–211, 2010.
- 447 [3] A. Becker, S. P. Fekete, P. Keldenich, M. Konitzny, L. Lillian, and C. Scheffer. Coordinated Motion
 448 Planning: The Video. In *Proceedings of the Symposium on Computational Geometry (SoCG)*, pages
 449 74:1–74:5, 2018. Video available at <http://www.computational-geometry.org>.
- 450 [4] A. T. Becker, E. D. Demaine, S. P. Fekete, G. Habibi, and J. McLurkin. Reconfiguring massive particle
 451 swarms with limited, global control. In *Proceedings of the International Symposium on Algorithms and
 452 Experiments for Sensor Systems, Wireless Networks and Distributed Robotics (ALGOSENSORS)*, pages
 453 51–66, 2013.
- 454 [5] A. T. Becker, E. D. Demaine, S. P. Fekete, and J. McLurkin. Particle computation: Designing worlds to
 455 control robot swarms with only global signals. In *Proceedings of the IEEE International Conference on
 456 Robotics and Automation (ICRA)*, pages 6751–6756, 2014.

- 457 [6] A. T. Becker, C. Ertel, and J. McLurkin. Crowdsourcing swarm manipulation experiments: A massive
458 online user study with large swarms of simple robots. In *Proceedings of the IEEE International Conference*
459 *on Robotics and Automation (ICRA)*, pages 2825–2830, 2014.
- 460 [7] A. T. Becker, O. Felfoul, and P. E. Dupont. Simultaneously powering and controlling many actuators
461 with a clinical MRI scanner. In *Proceedings of the IEEE/RSJ International Conference on Intelligent*
462 *Robots and Systems (IROS)*, pages 2017–2023, 2014.
- 463 [8] A. T. Becker, O. Felfoul, and P. E. Dupont. Toward tissue penetration by MRI-powered millirobots
464 using a self-assembled Gauss gun. In *Proceedings of the IEEE International Conference on Robotics and*
465 *Automation (ICRA)*, pages 1184–1189, 2015.
- 466 [9] A. T. Becker, G. Habibi, J. Werfel, M. Rubenstein, and J. McLurkin. Massive uniform manipulation:
467 Controlling large populations of simple robots with a common input signal. In *Proceedings of the*
468 *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 520–527, 2013.
- 469 [10] P. Berman and G. Schnitger. On the complexity of approximating the independent set problem.
470 *Information and Computation*, 96(1):77–94, 1992.
- 471 [11] S. Cannon, E. D. Demaine, M. L. Demaine, S. Eisenstat, M. J. Patitz, R. Schweller, S. M. Summers, and
472 A. Winslow. Two hands are better than one (up to constant factors). In *Proceedings of the International*
473 *Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 172–184, 2013.
- 474 [12] C. Chalk, E. Martinez, R. Schweller, L. Vega, A. Winslow, and T. Wylie. Optimal Staged Self-Assembly
475 of General Shapes. In *Proceedings of the European Symposium on Algorithms (ESA)*, pages 26:1–26:17,
476 2016.
- 477 [13] H.-L. Chen and D. Doty. Parallelism and time in hierarchical self-assembly. *SIAM Journal on Computing*,
478 46(2):661–709, 2017.
- 479 [14] E. D. Demaine, M. L. Demaine, S. P. Fekete, M. Ishaque, E. Rafalin, R. T. Schweller, and D. L. Souvaine.
480 Staged self-assembly: nanomanufacture of arbitrary shapes with $O(1)$ glues. *Natural Computing*,
481 7(3):347–370, 2008.
- 482 [15] E. D. Demaine, S. P. Fekete, P. Keldenich, H. Meijer, and C. Scheffer. Coordinated Motion Planning:
483 Reconfiguring a Swarm of Labeled Robots with Bounded Stretch. In *Proceedings of the Symposium on*
484 *Computational Geometry (SoCG)*, pages 29:1–29:15, 2018.
- 485 [16] E. D. Demaine, S. P. Fekete, C. Scheffer, and A. Schmidt. New geometric algorithms for fully connected
486 staged self-assembly. *Theoretical Computer Science*, 671:4–18, 2017.
- 487 [17] Z. Derakhshandeh, R. Gmyr, A. W. Richa, C. Scheideler, and T. Strothmann. An algorithmic framework
488 for shape formation problems in self-organizing particle systems. In *Proceedings of the Second Annual*
489 *International Conference on Nanoscale Computing and Communication (NANOCOM)*, page 21, 2015.
- 490 [18] Z. Derakhshandeh, R. Gmyr, A. W. Richa, C. Scheideler, and T. Strothmann. Universal coating for
491 programmable matter. *Theoretical Computer Science*, 671:56–68, 2017.
- 492 [19] M. Hoffmann. Motion planning amidst movable square blocks: Push-* is NP-hard. In *Canadian*
493 *Conference on Computational Geometry*, pages 205–210, 2000.
- 494 [20] P. S. S. Kim, A. T. Becker, Y. Ou, A. A. Julius, and M. J. Kim. Imparting magnetic dipole heterogeneity
495 to internalized iron oxide nanoparticles for microorganism swarm control. *Journal of Nanoparticle*
496 *Research*, 17(3):1–15, 2015.
- 497 [21] P. S. S. Kim, A. T. Becker, Y. Ou, M. J. Kim, et al. Swarm control of cell-based microrobots using a
498 single global magnetic field. In *Proceedings of the International Conference on Ubiquitous Robotics and*
499 *Ambient Intelligence (URAI)*, pages 21–26, 2013.

- 500 [22] A. V. Mahadev, D. Krupke, J.-M. Reinhardt, S. P. Fekete, and A. T. Becker. Collecting a swarm in a
501 grid environment using shared, global inputs. In *Proceedings of the IEEE International Conference on*
502 *Automation Science and Engineering (CASE)*, pages 1231–1236, 2016.
- 503 [23] S. Manzoor, S. Sheckman, J. Lonsford, H. Kim, M. J. Kim, and A. T. Becker. Parallel self-assembly
504 of polyominoes under uniform control inputs. *IEEE Robotics and Automation Letters*, 2(4):2040–2047,
505 2017.
- 506 [24] P. W. K. Rothmund and E. Winfree. The program-size complexity of self-assembled squares (extended
507 abstract). In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 459–468,
508 2000.
- 509 [25] M. Rubenstein, A. Cabrera, J. Werfel, G. Habibi, J. McLurkin, and R. Nagpal. Collective transport
510 of complex objects by simple robots: theory and experiments. In *Proceedings of the International*
511 *Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 47–54, 2013.
- 512 [26] H. M. Shad, R. Morris-Wright, E. D. Demaine, S. P. Fekete, and A. T. Becker. Particle computation:
513 Device fan-out and binary memory. In *Proceedings of the IEEE International Conference on Robotics*
514 *and Automation (ICRA)*, pages 5384–5389, 2015.
- 515 [27] S. Shahrokhi and A. T. Becker. Stochastic swarm control with global inputs. In *Proceedings of the*
516 *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 421–427, 2015.
- 517 [28] A. J. Thubagere, W. Li, R. F. Johnson, Z. Chen, S. Doroudi, Y. L. Lee, G. Izatt, S. Wittman, N. Srinivas,
518 D. Woods, E. Winfree, and L. Qian. A cargo-sorting DNA robot. *Science*, 357(6356):eaan6558, 2017.
- 519 [29] J. Werfel and R. Nagpal. Extended stigmergy in collective construction. *IEEE Intelligent Systems*,
520 21(2):20–28, 2006.
- 521 [30] J. Werfel and R. Nagpal. Three-dimensional construction with mobile robots and modular blocks. *The*
522 *International Journal of Robotics Research*, 27(3-4):463–479, 2008.
- 523 [31] E. Winfree. *Algorithmic self-assembly of DNA*. PhD thesis, California Institute of Technology, 1998.