Particle-Based Assembly Using Precise Global Control

Jakob Keller[1] · Christian Rieck[1] · Christian Scheffer[2] · Arne Schmidt[1]

Abstract

In micro- and nano-scale systems, particles can be moved by using an external force like gravity or a magnetic field. In the presence of adhesive particles that can attach to each other, the challenge is to decide whether a shape is constructible. Previous work provides a class of shapes for which constructibility can be decided efficiently when particles move maximally into the same direction induced by a global signal. In this paper we consider the single step model, i.e., a model in which each particle moves one unit step into the given direction. We restrict the assembly process such that at each single time step actually one particle is added to and moved within the workspace. We prove that deciding constructibility is NP-complete for three-dimensional shapes, and that a maximum constructible shape can be approximated. The same approximation algorithm applies for 2D. We further present linear-time algorithms to decide whether or not a tree-shape in 2D or 3D is constructible. Scaling a shape yields constructibility; in particular we show that the 2-scaled copy of every non-degenerate polyomino is constructible. In the three-dimensional setting we show that the 3-scaled copy of every non-degenerate polycube is constructible.

✉ Arne Schmidt
aschmidt@ibr.cs.tu-bs.de

Jakob Keller
jakobnkeller@gmail.com

Christian Rieck
rieck@ibr.cs.tu-bs.de

Christian Scheffer
christian.scheffer@hs-bochum.de

1   Department of Computer Science, TU Braunschweig, Braunschweig, Germany

2   Faculty of Electrical Engineering and Computer Science, Bochum University of Applied Sciences, Bochum, Germany

1 Introduction

In recent years, the easier access to micro- and nano-scale systems has given rise to challenges that deal with programmable matter. In some of these applications, particles can be controlled by a global external force such as gravity or a magnetic field. On actuation, every particle moves into the same direction at unit speed. Assembly of particles into desired structures using maximal movements, i.e., every particle moves into a given direction until it hits an obstacle or another particle, has been investigated in [2, 3, 11, 44]. However, it is also reasonable to expose the particles to these forces just for a limited amount of time, such that more precise movements become possible. Reconfiguration of a set of particles [3], gathering all particles [10, 35], or assembling patterned rectangles [15] are well studied problems.

In this paper, we consider the construction of tile-based structures (such as polyominoes in 2D, and polycubes in 3D) through adhesive particles, that all move one step into the same direction through activation of the global external force. We limit ourselves to the case where only one particle at a time is added to the assembling workspace at a predefined position. Whenever two particles come close, they stick together and no longer separate. This behavior is usually defined in terms of *glues* that are assigned to the particle's sides, and *temperature*. Glues are associated with a *binding strength*, so that two assemblies can only stick together, when the sum of binding strengths along their common boundary exceeds the temperature, e.g., see the introductory work of Winfree [55]. Several different glue types as well as attachment rules are defined in the literature (see Sect. 1.2). However, in this paper all particles have the very same glue on all sides, which is why we do not use this term any further and only argue that these particles stick to each other as soon as they are adjacent. To start the building process of a desired shape, we assume that there is a seed particle that is already placed in the workspace. Furthermore, we consider the seed to be "anchored" to its position so that it is not moved by the external force. This implies that only one particle actually moves at every time step.

In this model we consider the problem of deciding whether or not a given shape P is constructible. We call this the SINGLE STEP TILT ASSEMBLY PROBLEM (STAP). For all non-constructible shapes P we ask for a constructible subshape $P_{\max} \subset P$ of maximum size, i.e., $P_{\max}$ is the largest shape among all constructible subshapes of P. We call this optimization variant MAXSTAP.

For a precise model description and fundamental definitions, see Sect. 2.

1.1 Contributions

In this paper, we focus on the problem of constructibility within in the single step model in a free workspace. We prove that the problem is NP-complete in the three-dimensional setting (Theorem 7). The natural optimization variant allows for a $\Omega(n^{-1/d})$-approximation, where d denotes the dimension (Theorem 10). We provide a linear time algorithm that decides whether or not a given tree-shape is constructible (Theorem 13). In the case of non-constructible non-degenerate shapes, we show that the 2-scaled copy of every polyomino is constructible (Theorem 20), while

similar arguments show that the 3-scaled copy of every polycube is constructible (Theorem 24). The result is tight for the 2D case, and we conjecture that a 2-scaling also suffices for the 3D case.

1.2 Related Work

We divide the related work into three categories: Algorithmic results in tile self-assembly and tilt problems, and practical approaches.

1.2.1 Tile Self-Assembly

Instead of relying on universal, external control of agents, it is possible to use DNA as material. DNA-strands can either be used to fold into the desired shape [42], or to create building blocks based on *Wang tiles* [52, 53] that can then again self-assemble into shapes in a non-deterministic way. This was first introduced by Winfree [55] as the *abstract Tile Self-Assembly Model* (aTAM). The model became a standard in theoretical work on self-assembly, as well as proved suitable in practical experiments [41, 43, 45, 48, 56]. Instead of considering the case where all DNA tiles are binding to a seed assembly, it is also natural to consider multiple seeds that simultaneously grow subassemblies which can then again attach to each other. This 2-handed tile self-assembly model [18, 46] can also be used to let the subassemblies connect in phases, i.e., in a hierarchical manner [23]. One generalized version of hierarchical assembly is called "staged tile self-assembly" [19, 20, 24, 26]. Instead of relying on square-shaped tiles, some generalizations are also already considered [25, 29]. For more details on algorithmic self-assembly, we refer to the surveys by Doty [27], and Patitz [38], and Woods [57].

1.2.2 Tilt Problems

The process of self-assembly works by diffusion and is non-deterministic. If a deterministic approach is desired, we can use global control to move particles into different directions. Using global control opens a wide field of problems, that we summarize briefly.

Becker et al. [6] show that particles can be used for computation by implementing NOT, NAND, NOR, XOR and XNOR gates within a particularly designed obstacle environment. If the model is generalized by adding 2×1 particles (dominoes), it is also possible to construct FAN- OUT gates [9, 47].

Mahadev et al. [35] show how to gather n particles within an obstacle environment in $O(n^3)$ actuation steps. Becker et al. [10] improve the runtime to only depend on the geometric complexity of the workspace, rather than on the number of particles.

Closely related to this problem are occupancy, relocation and reconfiguration problems. For a given set of particles within an obstacle environment, these problems ask for a sequence of tilts such that (i) any particle reaches a designated position, (ii) a specific particle reaches a designated position, and (iii) every particle reaches its respective target position. If particles are allowed to move a unit step on actuation,

Caballero et al. [16] show that (i) permits a linear-time algorithm, while the decision variants of (ii) and (iii) are NP-hard. More recently, Caballero et al. [14] show PSPACE-completeness for (ii). Becker et al. [4] show that (i) is NP-hard when particles have to move maximally. Balanza-Martinez et al. [3] give tighter results by proving PSPACE-completeness for all three problems in the full tilt model. Note that these problems are hard, even if we do not allow the particles to stick together.

Manzoor et al. [36] provide algorithms for assembling shapes, called *drop shapes*, from *sticky* particles under global control. In this assembly process, only one particle at a time is added by maximal movements to a seed assembly. They also show that the assembly process can be pipelined, i.e., the same shape is produced multiple times. Becker et al. [11] prove that every drop shape permits an amortized construction time of $O(1)$, provided that sufficiently many copies are constructed. However, every shape needs a custom-designed obstacle workspace. Balanza-Martinez et al. [2] designed a single obstacle workspace in which every drop shape can be constructed, if it fits in a $w \times h$ rectangle. Additionally, Caballero et al. [17] describe a universal constructor for assembling shapes in the single step model. By using not only single particles but whole subassemblies, the class of constructible shapes increases, see [3, 44]. A crucial step is to decompose a given shape into two connected parts, that can then be pulled apart into a single direction, without causing collisions. Agarwal et al. [1] recently proved that this problem is NP-hard, even if a direction is given.

Another problem that arises from a practical point of view is error detection. Keldenich et al. [31] show that ortho-convex shapes can be classified within an obstacle workspace, i.e., these shapes can be tested for errors during the assembly process.

1.2.3 Practical Approaches

On the practical side, there are multiple approaches related to the assembly of microstructures with uniform movement. Related work on controlling a swarm of particles, especially in biological or medical environments, include control by electricity [13], chemical reactions [54], light [49, 54] and magnetism. Because magnetic resonance imaging (MRI) scanners already employ magnetism in a clinical context, it is reasonable to design a magnetic control device based on MRI scanners [7, 21, 37, 50, 51]. Another advantage of using MRI scanners lies in the possibility of tracking the positions of the agents as well as moving them. If the devices provide stronger custom coils, this approach might even lead to new applications where tissue penetration is wanted [8]. There are different types of particles discussed in the literature. On the one hand, there is research in which single cell organisms (*Tetrahymena Pyriformis*) are fed with iron particles so that they can be steered by applying a magnetic field [5, 33, 34]. On the other hand, research considers the fabrication of artificial micro-swimmers. These are usually called *Artificial Bacterial Flagella* (ABF) because they resemble their natural counterpart [28]. Another very popular approach is to design ABFs that consist of a magnetic head, that can be turned by a revolving magnetic field, and a rigid helical tail [22, 30, 39, 40, 58, 59]. By quickly turning the micro-swimmer, the tail generates a thrust force by its corkscrew shape.

2 Preliminaries and Problem Definition

In this section we provide a detailed description of the considered model as well as basic definitions that are used throughout the paper. If necessary, further definitions related to individual sections are given in the respective sections.

Polyomino. Let $P \subset \mathbb{Z}^2$ be a finite set of n grid points in the plane. The grid points of $\mathbb{Z}^2$ are called *positions*. The embedded graph G_P is the grid graph induced by P, in which two vertices are adjacent if they are at unit distance. If G_P is connected, we obtain a *polyomino* by placing a unit square, called *tile*, centered on every vertex of G_P. We refer to the *size* of a shape by the number of its tiles. At some points, we also use $|P|$ to denote the size of a shape P, if this is more practical. Two tiles or positions are *adjacent*, if their respective vertices in G_P are adjacent, i.e., they share an edge. A position $p \in \mathbb{Z}^2$ is *occupied*, if there is a tile that is placed on p, and *free* otherwise. The neighborhood $N[\cdot]$ of a tile or position is the respective set of adjacent positions. A polyomino P is *simple* if the grid graph $\mathbb{Z}^2 \backslash G_P$ is connected. A polyomino is *degenerate* if there exist two tiles $t_1, t_2 \in P$ such that $|N[t_1] \cap N[t_2]| = 2$, and each position $p \in N[t_1] \cap N[t_2]$ is free. If G_P is a tree, the respective polyomino is *tree-shaped*.

Workspace. The *workspace* is a rectangular region of $2n \times 2n$ positions, anchored at position $(0, 0)$. The size of the workspace is proportional to the size of the shape to be constructed, so that there is enough space to all sides. The workspace contains an *anchored seed tile* at position (n, n). Note that the term "anchored" does not require that the seed tile has a specific position within the shape, but only that it occupies a specific position within the workspace during the whole construction process. Thus, the seed actually is not affected by the global force, i.e., it does not move. So all tiles connected to the seed will not move either.

Construction step. A tile can *move* from one position p to an adjacent position q as long as the neighborhood $N[p]$ is free. A *construction step* is a sequence σ of moves such that σ moves a tile to a position adjacent to an occupied position, i.e., adjacent to a tile of the current assembly. Every construction step starts by *adding* a new tile to the workspace. Without loss of generality, we assume that every construction step starts at position $(0, 0)$. Note that we limit a construction step to be completed, before the subsequent one can start. This means that during a construction step, only the newly added tile moves within the workspace. Whenever two tiles are at adjacent positions, they stick together and no longer separate. Analogously, we define a *deconstruction step* as a sequence $\widetilde{\sigma}$ that moves a tile from its position to the position $(0, 0)$. Note that these are "reverse" moves, i.e., a tile can move from a position p to an adjacent position q if $N[q]$ is free. If there is a deconstruction step for a tile t, we call t *removable*.

   We have not described the actual process of adding a tile to the workspace, as we do not use this in a real-life application. However, one idea would be to place all needed particles at a sufficiently large distance from each other so that they do not get too close to each other during a construction step. With this, one could make sure that a single particle is available at a predefined position after a construction step. Another possibility would be to keep them in a bin in which they cannot stick to each other, i.e.,

the glue only really sticks when particles are on the workspace. To add a new particle to the workspace, for example, a physical gate would have to be opened.

Constructibility. Beginning with a seed tile, a polyomino P consisting of n tiles is constructible, if and only if there is a *construction sequence* $\Sigma = (\sigma_1, \sigma_2, \ldots, \sigma_{n-1})$ of $n-1$ consecutive construction steps such that the resulting polyomino P', induced by successively adding tiles with Σ, is identical to P. Reversing Σ yields a *deconstruction sequence* $\widetilde{\Sigma}$, i.e., a sequence of deconstruction steps that iteratively removes tiles from P, eventually resulting in the empty workspace. We call a shape *indestructible*, if there exist no deconstruction sequence for that shape.

Problem description. With the definitions given above, we are interested in the constructibility for a given shape. In particular, we consider the following decision problem.

SINGLE STEP TILT ASSEMBLY PROBLEM (STAP)

Given a shape P of size n and an anchored seed tile, does there exist a construction sequence of $n - 1$ consecutive construction steps that constructs P?

For non-constructible shapes we are interested in the maximum sized subshape that can be constructed. So we consider the following optimization problem.

MAXIMUM SINGLE STEP TILT ASSEMBLY PROBLEM (MAXSTAP)

Given a non-constructible shape P of size n and an anchored seed tile, construct a subshape $P_{max} \subset P$ where $|P_{max}|$ is maximized.

If the problems described are considered in a certain dimension, we specify this as a prefix of the respective problem name, e.g., 2D-STAP describes STAP in the two-dimensional case.

Because we will use the fact that construction and deconstruction are basically the same problems throughout almost all arguments given in the paper, we want to restate the following result by Becker et al. [11] for the full tilt model.

Theorem 1 (Theorem 2, [11]) *A polyomino P can be constructed if and only if it can be deconstructed using a sequence of tile removal steps that preserve connectivity. A construction sequence is a reversed deconstruction sequence.*

To see that this is indeed true, consider a single construction step. The tile that is added to the assembly in this particular step can eventually be removed in a deconstruction of the shape on the exact same path. It is easy to see that this result can be adapted for the particular single step model considered in this paper.

Note that the definitions from above are given for the 2D setting. It is straightforward to extend these to the 3D setting, by letting $P \subset \mathbb{Z}^3$, introducing two additional directions in that a tile can move, and considering unit cubes instead of unit squares as tiles, and anchoring the workspace, that contains a seed tile at position (n, n, n), at position $(0, 0, 0)$.

Furthermore, it is easy to see that Theorem 1 applies in both models to three dimensions.
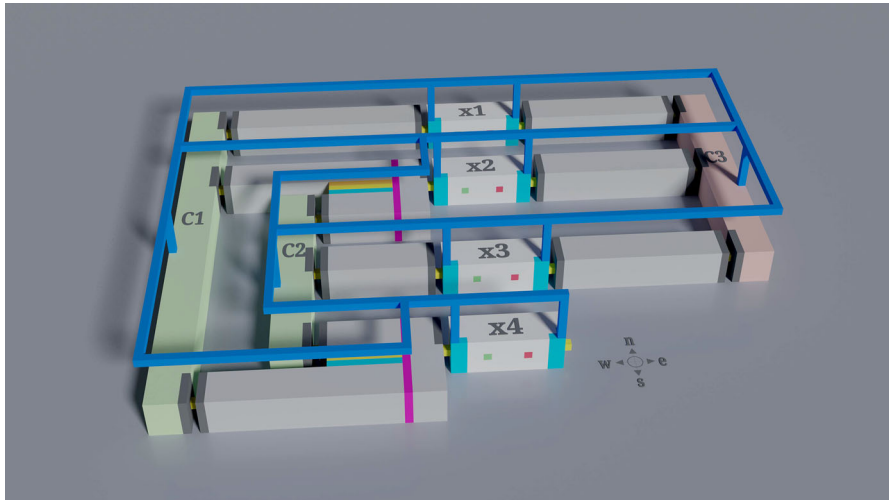
Fig. 1 Symbolic overview of the NP-hardness reduction. The reduction is based on the NP-hard problem PLANAR MONOTONE 3SAT. The depicted instance is due to the formula $\varphi = (x_1 \lor x_2 \lor x_4) \land (x_2 \lor x_3 \lor x_4) \land (\overline{x_1} \lor \overline{x_2} \lor \overline{x_3})$. The variable gadgets are shown in white, while the unnegated and negated clause gadgets are shown in green and red, respectively. The gray cuboids represent connector gadgets. The 'colorful lines' are representing conjunction gadgets. All clauses and variables are connected by a blue frame above the construction to guarantee connectivity during a feasible deconstruction (Color figure online)

3 3D-STAP Is NP-Complete

Becker et al. [11] showed that it is NP-complete to decide whether or not a polycube is constructible in the full tilt model. However, in our model, the paths that add new tiles to the assembly can be more complex than just a straight line, and thus we cannot simply adapt their proof. In this section, we show that 3D-STAP is NP-complete. The proof is based on a reduction from the NP-hard problem PLANAR MONOTONE 3SAT [12]. This problem asks to decide whether a Boolean 3-CNF formula φ is satisfiable, for which in each clause the literals are either all unnegated or all negated, and for which the clause-variable incidence-graph is planar. Because of Theorem 1, we will argue that a polycube is deconstructible if and only if the corresponding Boolean formula φ as an instance of planar monotone 3-sat is satisfiable.

3.1 Outline of The NP-Hardness Reduction

For every instance φ of PLANAR MONOTONE 3SAT, we construct a polycube P_φ as an instance of 3D-STAP. We consider a rectilinear planar embedding of the variable-clause incidence graph G_φ of φ where the variable vertices are placed on a line, and clauses containing unnegated and negated literals are placed on either side, respectively. For a schematic overview of P_φ, consider Fig. 1. In P_φ, each variable and each clause of φ is represented by a variable gadget and a clause gadget, respectively. To realize the fact that a variable can be contained in several clauses, we introduce a
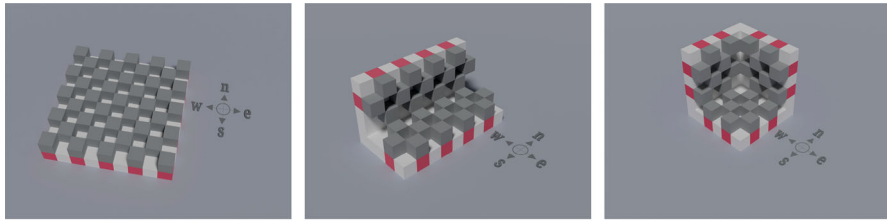
(a) Indestructible 1-wall.   **(b)** Indestructible 2-wall.   **(c)** Indestructible 3-wall.

Fig. 2 Indestructible walls. Red cubes indicate the respective positions of teeth, which in turn are shown in dark gray. Due to the design of these polycubes, a tooth must first be removed before the associated red cube can be removed from the assembly; otherwise the connectivity is lost (Color figure online)

conjunction gadget. An edge in G_φ is realized by a connector gadget. To guarantee connectivity during the deconstruction of the polycube P_φ, we need to make sure that parts of the variable gadgets that are not participating in the satisfying assignment, are not disconnected in several parts. Therefore, we add a frame above the actual polycube, connecting all clauses with certain parts of the variable gadgets.

We can show that there is a deconstruction sequence for P_φ if and only if φ is satisfiable. By using a checkered tile arrangement within all gadgets, we can enforce a specific deconstruction sequence. On the one hand, we ensure that, due to the connectivity constraint, either the part of the variable that is connected to their unnegated or to their negated literal containing clauses can be deconstructed; thus, these deconstruction steps can be used to determine a valid variable assignment for φ. This implies that, together with the conjunction gadgets, all clauses containing the respective literal can be deconstructed. On the other hand, the other side of the variable gadget can only be deconstructed, if all other clauses are already deconstructed, i.e., if the clauses are satisfied by other variable assignments.

3.2 Construction of The Gadgets

In the following, we describe several polycubes that serve as gadgets in the NP-hardness reduction. In particular, we need polycubes for variables and clauses, as well as for (logic) conjunction. Furthermore, we need a gadget to connect several gadgets with each other, i.e., a connector gadget. All these gadgets are based on the following polycube that cannot be deconstructed if we restrict the deconstruction direction.
Indestructible Wall. A *wall* is the polycube depicted in Fig. 2(a). It consists of two layers, an odd sized *solid layer*, and a checkered *tooth layer*. This tooth layer consists of non-adjacent cubes (shown as dark gray cubes) at even positions. This construction can simply be modified to construct a *k-wall*, see Fig. 2 for examples. Note that k can be at most 6, and there exist several *k*-walls for $k \in \{3, 4\}$.

Recall that a deconstruction step is a sequence of moves that moves a tile of the polycube to position $(0, 0, 0)$. The crucial property of the wall is that it is not deconstructible *from its solid layers*. First, we want to give an intuitive idea why this is true for the 1-wall. For this, assume for a moment that the 1-wall partitions the workspace into two parts, i.e., any tile in the top part of the workspace cannot pass the wall side-
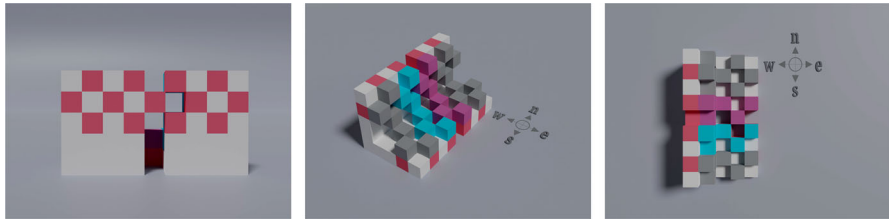
Fig. 3 Different views on a disconnected 2-wall. To separate the 2-wall into two disconnected components, light gray cubes of the solid layer are removed in the space between the blue and purple teeth (Color figure online)

ways to reach the bottom part, and vice versa. Furthermore, the solid layer and the position $(0, 0, 0)$ lay in the bottom part. It is clear that we need to remove a tile from the solid layer first. But no matter which one we choose, all its neighbors cannot be removed, because of the connectivity restriction. So, we cannot remove any tile from the checkered tooth layer.

With the assumption, that we only have access to the solid layers, we obtain the following.

Lemma 2 *A k-wall, $k \in \{1, \ldots, 6\}$, is not deconstructible from its solid layers.*

Proof Suppose for the sake of a contradiction that a k-wall is deconstructible from its solid layers, and let $\widetilde{\Sigma} = (\sigma_1, \ldots, \sigma_n)$ be a deconstruction sequence. Because we only have access to the solid layers, σ_1 clearly has to remove a cube from a solid layer. We partition the solid layer in cubes at even and odd positions. Because there are cubes in the tooth layer at even positions, we can only remove cubes at odd positions from the solid layer to maintain connectivity. But then there cannot be a $\widetilde{\sigma}_i \in \widetilde{\Sigma}$ such that $\widetilde{\sigma}_i$ removes a cube from the tooth layer. This is a contradiction to the existence of $\widetilde{\Sigma}$. Thus, a wall is not deconstructible from the solid layer. □

We show that the specific design of the tooth layers of a k-wall is necessary to guarantee the indestructibility, i.e., if at least one tooth is missing, the resulting polycube is deconstructible from the solid layer.

Lemma 3 *There is at least one position p at a tooth layer of a k-wall, $k \in \{1, \ldots, 6\}$ such that the k-wall is deconstructible from the solid layer if p is free.*

Proof Let p be the position of the missing tooth, and consider the respective position in the solid layer. Removing this tile and the four adjacent tiles will yield a hole that is large enough that tiles can pass through it. Thus, tiles from the tooth layer can be removed. To do so, we position a tile from the tooth layer above p and move it in the direction of the hole such that there will be no face contact to other tiles. Thus, all teeth can be removed one by one, followed by deconstructing the remaining parts of the solid layer. □

A simple observation is that these k-walls can arbitrarily be enlarged without losing the property of being indestructible. Of particular interest for the reduction are enlarged 6-walls, called *cuboids*, that will serve as *clause* and *connector gadgets*.
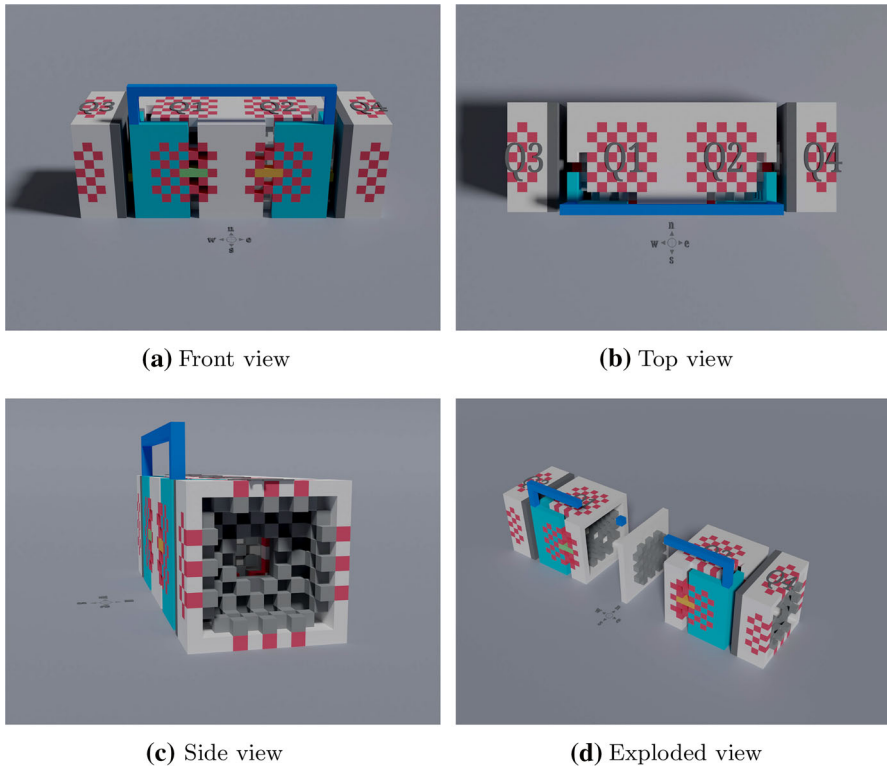
(a) Front view

(b) Top view

(c) Side view

(d) Exploded view

Fig. 4 Different views on the variable gadget. The actual variable gadget consists of the indestructible cuboids Q_1 and Q_2, that are modified in such a way that there are L-shaped parts (shown in light blue) that are only connected to the remaining assembly via the green and orange bridges. The dark blue tiles indicate a part of the connectivity frame (Color figure online)

Another crucial observation is that because a k-wall is not deconstructible from its solid layers, we can leave out several cubes of the solid layers so that the remaining shape is disconnected into two parts, see Fig. 3 for an example of a disconnected 2-wall. This insight will lead to a configuration that allows for a decision, i.e., that will serve as the variable gadget.

If two cuboids have to be connected, we place them at distance one to each other and add a single cube to connect them. Furthermore, we remove all cubes in a 3×3 area at matching sides such that we can move cubes from the inside of one cuboid to the other through these holes, see Figs. 4(c) and 4(d) for illustration.

Variable Gadget. The variable gadget consists of two indestructible cuboids (Q_1 and Q_2) that share a solid layer, see Fig. 4(d) for an exploded illustration. As shown in Fig. 4, we remove tiles (similar to Fig. 3) to separate an L-shaped part of each cuboid (light blue tiles). These shapes are then reconnected by two bridges (green and orange tiles), see Fig. 4(a). Additionally, the L-shaped parts are connected by a thin frame above the cuboids (dark blue tiles).
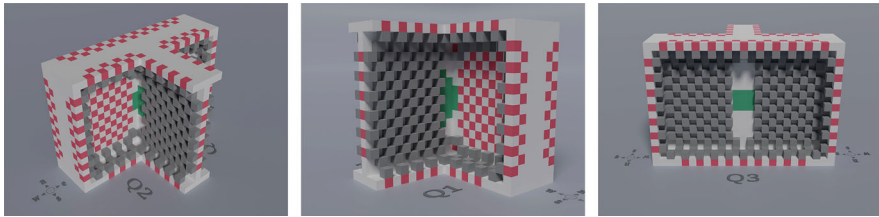
Fig. 5 Different views on the conjunction gadget. At the indicated positions, three indestructible cuboids Q_i are attached. The whole construction is deconstructible, if and only if both, Q_1 and Q_2, or Q_3 is deconstructible. This follows from the design of the respective teeth layers, as well as the dark green tiles (Color figure online)

Observation 4 Solely removing the green and orange tiles of a variable gadget results in a disconnected shape.

As a consequence of Observation 4, the forced choice of removing either the green or the orange tiles, can be used to determine an assignment for the respective Boolean variable. It remains to show how a variable gadget can be deconstructed, if additional cuboids are attached at each side.

Lemma 5 *Let P be a polycube that is put together by a variable gadget and one cuboid* (Q_3 *and* Q_4) *at each end, connected to the respective L-shaped parts. Then P is only deconstructible if at least* Q_3 *or* Q_4 *is deconstructible.*

Proof Without loss of generality, let Q_3 be deconstructible. Then the orange tiles can be removed in order to deconstruct Q_4 (by Lemma 3), and Q_2 afterwards. Due to the hole between Q_1 and Q_3 and the assumption that Q_3 is deconstructible, Q_1 can also be deconstructed.

On the other hand, if neither Q_3 nor Q_4 is deconstructible, only the green or the orange tiles can be removed (by Observation 4). But then, either Q_3 or Q_4 can be deconstructed, but not both, resulting in an indestructible shape. □

As the last ingredient for our NP-hardness reduction we need a gadget that realizes a conjunction. This gadget will be used to guarantee that a variable gadget can be completely deconstructed if and only if all clauses in which the respective variable participates are satisfied.

Conjunction Gadget. As illustrated in Fig. 5, the conjunction gadget is T-shaped. The wall between the cuboids Q_1 and Q_2 contains teeth to both sides, whereas the wall at cuboid Q_3 has teeth except for the positions where the T-shape is connected. This connection will be the crucial part to deconstruct this gadget. At all three positions (Q_1, Q_2, and Q_3) we attach connector gadgets leading either to another conjunction, to a variable, or to a clause gadget. Note that these connector gadgets have the same size as the conjunction gadget, i.e., the solid layers of the connectors and the conjunction gadget match.

We can show that this gadget is deconstructible if and only if the cuboid at Q_3, or both cuboids at Q_1 and Q_2 are deconstructible.

Lemma 6 *Let P be a polycube that is put together by three cuboids Q_1, Q_2, and Q_3 which are connected by a conjunction gadget. Then P is deconstructible if and only if Q_1 and Q_2 are both deconstructible, or Q_3 is deconstructible.*

Proof For the proof, we distinguish two cases.

First, let Q_1 and Q_2 be deconstructible. Thus, all teeth from the respective insides can be removed. After this, the wall that is shared by Q_1 and Q_2 becomes deconstructible. Afterwards, by removing the dark green tiles (see Fig. 5), we create a hole to reach the inside of Q_3 that makes Q_3 deconstructible as well. Thus, P is deconstructible.

Now assume that either Q_1 or Q_2 is deconstructible, but not both. Without loss of generality consider Q_1 to be deconstructible. Then all teeth from its inside can be removed. By this, neither Q_2 nor Q_3 become deconstructible because the teeth on the respective sides do not permit the removal of enough tiles such that tiles from the respective insides can pass through. Thus, P is not deconstructible.

It remains to show that P is deconstructible if Q_3 is deconstructible. Because Q_3 is deconstructible, we can remove the teeth at this side of the conjunction gadget. Because there are no teeth at the opposite sides, the whole wall can be deconstructed. This results in holes to the inside of Q_1 and Q_2 such that these cuboids become deconstructible as well. □

By putting all these together, we obtain the following.

Theorem 7 3D- *STAP is NP-complete.*

Proof First we note that the problem of deciding whether or not a polycube is constructible is in NP. For this, we guess a permutation of the involved tiles and check if there is a feasible construction step for each tile. It is easy to see that a single construction step has length $O(n)$. Because the target polycube has size n, a construction sequence has length $O(n^2)$.

To show NP-hardness, consider a rectilinear planar embedding of the variable-clause incidence graph G_φ of a given PLANAR MONOTONE 3SAT formula φ, where the variable vertices are placed horizontally in a row, and clauses containing unnegated and negated literals are placed above and below this row, respectively. We place a variable gadget for every variable (white in Fig. 1), and a cuboid for each clause (green in Fig. 1). These blocks are connected via connector cuboids (gray) and conjunction gadgets whenever the number of unnegated or negated occurrences of a variable is larger than one. On top of this construction, we use a frame to hold all clauses and L-shaped parts in the variable gadgets together. This will be necessary to deconstruct the shape completely whenever the underlying Boolean 3Sat formula is satisfiable.

Claim 8 *If there is a deconstruction sequence $\widetilde{\Sigma}$ for P_φ, then there is a satisfying assignment for φ.*

In order to deconstruct a clause gadget, at least (parts of) one of its literal containing variable gadgets has to be already deconstructed. Thus, every deconstruction sequence has to begin with deconstruction steps that remove either the green or orange tiles from any variable gadget. As argued in Observation 4, the green and orange tiles cannot
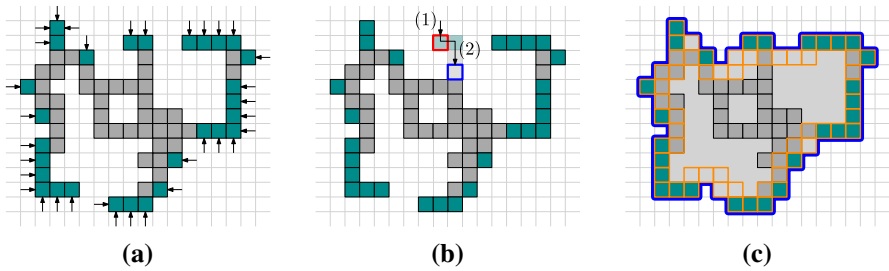
Fig. 6 (**a**) Boundary tiles (dark cyan) and non-boundary tiles (gray). (**b**) Every step of the algorithm adds a tile in two steps: (1) Move a new tile t to a boundary position p of P that is free. (2) Move t on P from p to a position that is adjacent to a tile of the current polyomino P'. (**c**) The curve B (blue), and the set T (orange) (Color figure online)

both be removed, i.e., setting a variable to true and false simultaneously is not possible. Because $\widetilde{\Sigma}$ is a deconstruction sequence, eventually each clause will be deconstructed.

Therefore, there is a satisfying assignment for φ given by the order in which each variable of P_φ is deconstructed by $\widetilde{\Sigma}$. In particular, for each $x_i \in \varphi$, we set $x_i = 1$ if the respective green tiles are removed first, and $x_i = 0$ otherwise. □

Claim 9 If the Boolean formula φ is satisfiable, then P_φ is deconstructible by some deconstruction sequence $\widetilde{\Sigma}$.

Let α be a satisfying assignment of φ. Then the polycube P_φ can be deconstructed as follows: According to whether a variable is set to true or false in α, either the green or orange tiles are removed, respectively. These deconstruction steps produce holes, large enough for a deconstruction of the whole literal representing cube, as argued in Observation 4. Afterwards, because there is a hole from the clause cuboids to the literals, all clauses that contain these literals can be deconstructed by Lemma 3. Because α is a satisfying assignment for φ, all clause gadgets and the respective parts of the variable gadgets can be deconstructed. The respective parts of the variable gadgets that are not participating in the satisfying assignment are connected by the overlaying connectivity frame. Because the clauses are already deconstructed, there are holes through which the remaining parts of the variable gadgets can be deconstructed. Removing the connectivity frame results in deconstructing P_φ.

These two claims complete the proof. □

4 Optimization Variant and Approximation

For polyominoes and polycubes that cannot be constructed, it is natural to consider the problem of constructing a subshape of maximum size. We show that for each shape P of size n in dimension d, a portion of $\Omega(n^{(d-1/d)})$ can always be constructed, implying an $\Omega(n^{-1/d})$-approximation for MAXSTAP.

Theorem 10 *In dimension $d = 2, 3$, the greedy algorithm is an $\Omega(n^{-1/d})$-approximation for* MAXSTAP.

(a)   **(b)**

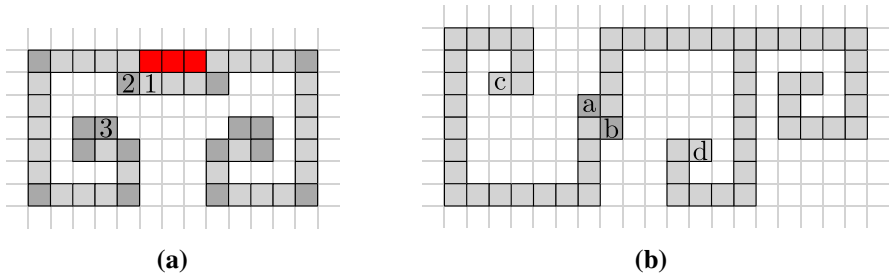**Fig. 7** (**a**) No corner tile (dark gray) can be removed, because they either do not have a deconstruction step or are essential for connectivity. Successively removing the tiles 1, 2, and 3 by suitable deconstruction steps, the obtained shape can easily be deconstructed. Removing the red tiles first results in an indestructible shape. (**b**) By removing a first, we can remove the spiral starting with c and afterwards the rest. By removing b first, the spiral starting with d can be removed, but the remaining shape is indestructible (Color figure online)

Proof We prove the theorem by showing that greedily filling up accessible free positions leads to a polyomino $P' \subseteq P$ with $|P'|/|P| \in \Omega(n^{-1/d})$. A position p is *accessible* with respect to a polyomino P if we can move a tile t to this position such that t is never adjacent to a tile of P unless t lies on p. A tile $p \in P$ is a *boundary tile* of P if p is accessible with respect to $P \setminus p$, see Fig. 6(a). If there is a boundary tile p of P that is not part of P', we add a new tile t to P' in two steps, see Fig. 6(b): (1) We move t to the position p. (2) We move t on P to a position adjacent to a tile of P'. This implies that the greedy algorithm ends up with a polyomino holding all boundary tiles of P.

Next, we show a lower bound on the boundary tiles of P of $\Omega(n^{(d-1)/d})$. We will only show this for dimension $d = 2$, similar arguments hold for $d = 3$. Let B be the union of all edges lying between an accessible and a non-accessible position with respect to P, see blue curve in Fig. 6(c). B is a non-self-intersecting curve by the definition of accessible positions. Thus, B partitions the plane into a bounded area A containing P and an unbounded area. Let T be the union of all positions from A sharing at least a corner with B, see the orange positions in Fig. 6(c). Then $|T| \geq \sqrt{|A|}$. Note that not each position of T is occupied by P, see the light gray positions in Fig. 6(c). Let T' be the positions along T that share an edge with B. It is easy to see that $2|T'| \geq |T|$. Each position p from T' that is not a boundary tile from P is adjacent to a boundary tile p' from P. We call p' a *blocking tile* of p. Each boundary tile is a blocking tile for a constant number of positions $p \in T'$ that are not a boundary tile. Hence, there are $\Omega(|T'|) = \Omega(|T|)$ many boundary tiles. Because $P \subseteq A$, we obtain $|P| \leq |A|$ implying $|T| \in \Omega(\sqrt{|A|}) \subseteq \Omega(\sqrt{|P|})$. Hence, the shape P' constructed by the greedy algorithm has at least $\Omega(n^{1/2})$ boundary tiles. A similar surface-to-volume argument implies a lower bound of $\Omega(n^{2/3})$ in the three-dimensional case.

Therefore, in dimension $d = 2, 3$, this approach yields an approximation factor of $|P'|/|P| = \Omega(n^{-1/d})$.   □

5 Efficient Algorithms for Special Classes of Shapes

Due to the equivalence of construction and deconstruction, we pursue the goal of classifying tiles that can be removed from a shape, without harming its deconstructibility. In general, this seems to be really difficult. On the one hand, it is not sufficient to restrict the search for removable tiles to *corners* (tiles with exactly one horizontal and one vertical neighbor), because for successfully deconstructing a polyomino, it may be necessary to remove non-corner tiles first, see Fig. 7(a). On the other hand, removing non-corner tiles can result in an indestructible subshape, again see Fig. 7(a). Furthermore, even in simple polyominoes, the removal of a corner tile can result in an indestructible subshape, see Fig. 7(b). Note that the latter is not the case in the full tilt model [11].

Therefore we consider special classes of shapes and show that the problem becomes easy in the case of trees and scaled shapes.

5.1 Tree Shapes

We show that STAP can be decided in linear time (in the size of the shape) for the class of tree-shapes by a greedy algorithm. We show this in detail for the two-dimensional setting, i.e., for tree-shaped polyominoes. The results can simply be adapted for tree-shaped polycubes in three dimensions. Because the removal of a tile with more than one neighbor results in splitting the polyomino in several parts, we are restricted to remove tiles with exactly one neighbor, i.e, leaves. If there are any tiles left, but no further tile can be removed, we conclude that the polyomino cannot be constructed.

We begin by stating two facts about removable tiles. Firstly, by removing a removable tile, other tiles do not lose their property of being removable. And secondly, if a tree-shaped polyomino is constructible, then after removing any removable tile, the resulting polyomino is also constructible.

Lemma 11 *Let P be a tree-shaped polyomino and $\mathcal{R}_P$ the set of removable tiles of P. For all $P' \subseteq P$ it holds that if $t \in \mathcal{R}_P \cap P'$, then $t \in \mathcal{R}_{P'}$.*

Proof Let $t, t' \in \mathcal{R}_P$ and $P' := P \setminus t'$. Due to the definition of $\mathcal{R}_P$, P' is connected. Furthermore, the path that removes t from P still exists, because the removal of t' cannot block this path. Therefore, $t \in \mathcal{R}_{P'}$. □

Lemma 12 *Let P be a constructible tree-shaped polyomino and let t be a removable tile. Then $P \setminus t$ is also constructible.*

Proof For the sake of a contradiction, let $P' \subsetneq P$ be the largest polyomino that results by removing tiles from P such that $t \in P'$ and $P' \setminus t$ is constructible. Let t' be the last tile that was removed in this process to obtain P' from P. Because of Lemma 11, a feasible deconstruction sequence can contain two subsequent deconstruction steps so that the first one removes t from the shape $P' \cup t'$, followed by removing t'. Thus, P' was not the largest constructible polyomino, which is a contradiction. □
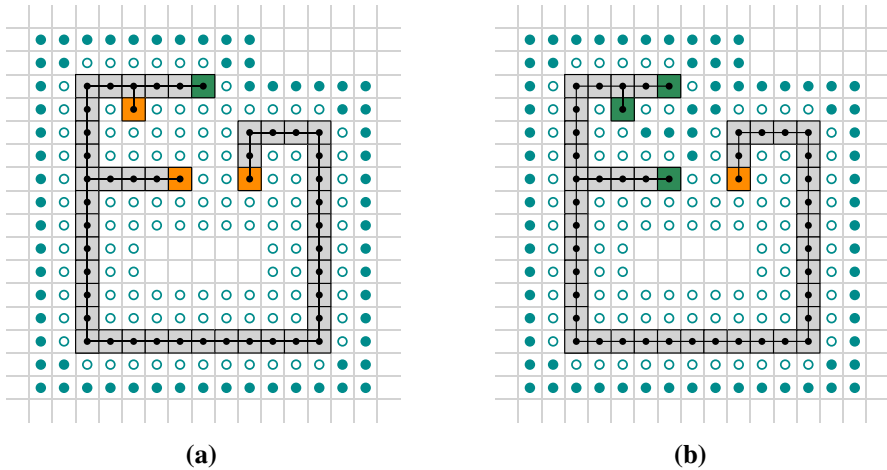
Fig. 8 (**a**) A polyomino P (gray) and the dual graph of P (black). Tiles in green are removable, and tiles in orange are leaves that are not removable yet. Cyan dots represent free positions with the state `unblocked`, cyan circles represent free positions with the state `blocked`. (**b**) The situation after the removal of the green tile in (a) (Color figure online)

By using Lemma 12 iteratively, we obtain a simple strategy that decides whether a tree-shaped polyomino is constructible or not. By applying suitable subroutines and data structures, this yields a linear-time algorithm.

Theorem 13 *Let P be a tree-shaped polyomino of size n. It can be decided in $O(n)$ time whether or not P is constructible.*

Proof Notice that every free position within the workspace that has an L_∞-distance greater than 2 to any tile of the shape has no effect on a potential indestructibility of that shape. Thus, to keep track of the tiles that are and that become removable, we just have to consider every position within an L_∞-distance of at most 2 to any tile of the shape. These positions are marked as `blocked`. Note that there are at most $O(n)$ `blocked` positions, because each tile has at most 24 positions in a distance of at most 2.

Blocked positions that are not adjacent to P, but are reachable from the outside of its bounding box are marked as `unblocked`. This can be done by a simple graph scan algorithm like BFS or DFS in $O(n)$ time.

To identify whether a tile is removable, we only need to check the following: (i) t is a leaf tile, and (ii) at least one position of $N[t]$ is adjacent to an `unblocked` position. This procedure costs $O(1)$ time per tile; thus $O(n)$ time in total.

The following steps are executed iteratively, until either all tiles have been removed, or no further tile can be removed from P. For this, let $t \in P$ be a removable tile.

1. Remove t from P and mark its former position as `blocked`.
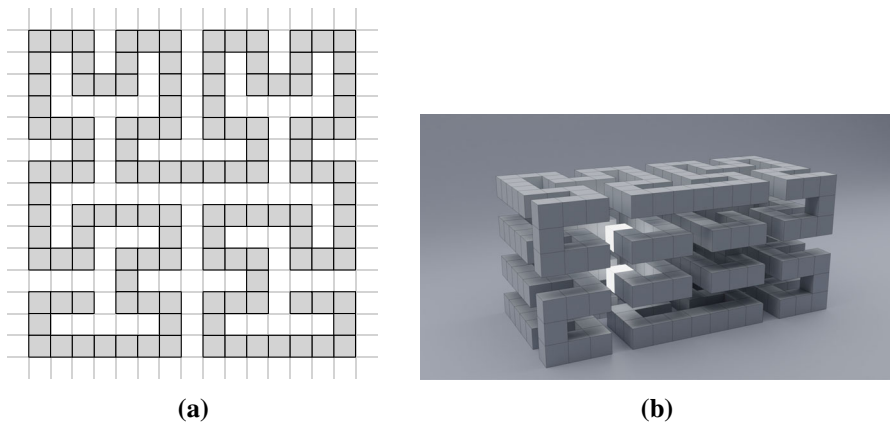2. For each free position $p \in N[t]$ that is adjacent to an `unblocked` position:

Fig. 9 (**a**) A (slightly modified) 2D Hilbert curve that cannot be constructed. (**b**) A (slightly modified) 3D Hilbert curve that cannot be constructed; the two leaves are represented as luminous cubes

(a) Mark p as `unblocked` if $N[p]$ is free and start a graph scan on `blocked` positions as well as mark them as `unblocked`, if they are reachable from p and their respective neighborhood is free.

(b) During the scan, if a `blocked` position does not change its state, check whether adjacent tiles become removable.

If there are tiles left that cannot be removed, we conclude that P is not constructible. Otherwise, P is constructible and the output is a feasible construction sequence.

Each free position can change its state only once. We charge the constant cost of the checking part in Step 2(b) of the algorithm to the respective adjacent `unblocked` position. Thus, over all iterations, each position is charged $O(1)$ cost, yielding a runtime of $O(n)$.   □

It is easy to see that the same holds true for tree-shapes in 3D.

Corollary 14 *Let P be a tree-shaped polycube of size n. It can be decided in $O(n)$ time whether or not P is constructible.*

5.2 Scaled Shapes

In the previous section we showed that it can be decided efficiently whether or not a tree-shape is constructible. A crucial point is that these shapes are thin. A simple example for a non-constructible tree-shape is a slightly modified *Hilbert curve* (in fact it is even a path) in that both endpoints are not removable, see Fig. 9. Therefore, we want to assume that we are allowed to construct a *scaled copy* of the actual shape.

Definition 15 Let P be a polyomino and $c \in \mathbb{N}$. By P^c we denote the c-scaled copy of P, i.e., each tile in P is replaced by a $c \times c$ square of tiles.

We show that the 2-scaled copy of a non-degenerate polyomino is constructible. Recall that in a non-degenerate polyomino P, for every non-adjacent pair $t_1, t_2 \in P$ of

Fig. 10 For deconstruction, it is necessary to remove at least one of the red tiles. Independent from the scaling factor, both tiles block each other and we are only able to deconstruct a staircase to the right and below both tiles (hatched part) (Color figure online)
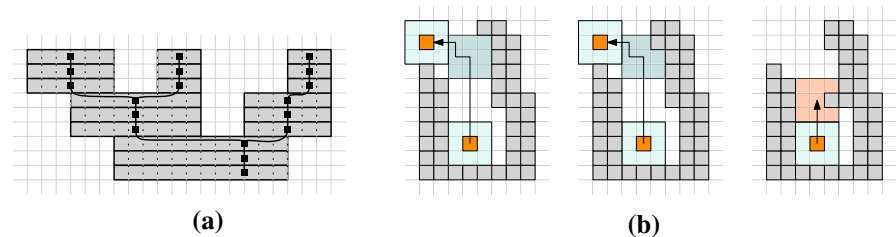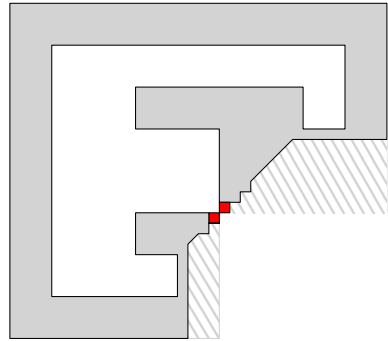

(a)   **(b)**

Fig. 11 (**a**) The 3-scaled copy P^3 of a polyomino P (gray tiles), its partition into horizontal slabs $\mathcal{S}_P$, and its dual graph $\mathcal{C}(\mathcal{S}_P)$ (dark gray). (**b**) A polyomino that is 3-empty, weakly 3-empty, and not (weakly) 3-empty, respectively (Color figure online)

tiles with $|N[t_1] \cap N[t_2]| \neq \emptyset$, there is at least one occupied position $p \in N[t_1] \cap N[t_2]$, i.e., the tiles t_1, t_2 have a common neighbor. Note that no scaling factor suffices to guarantee constructibility for degenerate polyominoes, see Fig. 10.

Definition 16 We call a polyomino P *c-empty*, if for every pair (p_i, p_j) of free positions in the same connected component of $G_{\mathbb{Z}^2 \setminus P}$ there is a square of size $c \times c$ that initially overlaps with p_i can be moved in such a way that it eventually overlaps with p_j without overlapping with any position of P at any time. We call P *weakly c-empty* if at any time at most one corner of the square overlaps with a position of P.

For an illustration of Definition 16, see Fig. 11(b). Note that in the case of weakly 3-emptiness it is sufficient to ensure that during the motion of a tile its neighborhood $N[\cdot]$ is kept empty. However, it is not possible to limit the definition of weakly 3-emptiness to this, because then free positions that are adjacent to two tiles (especially "corner positions" in the free space) would be excluded from this definition. For a 3-empty polyomino P, it is straightforward to see that as long as a tile $t \in P$ can be moved to a free position p that lies in the outer face, such that all surrounding positions of p are free as well, t can be removed from P. The intuition is the following: Consider a path of a 3×3-square, centered at that free position p, that connects p to the outside of the bounding box of P. The deconstruction step for the tile t consists of all positions given by that path. Note that this still holds if we consider weakly 3-empty polyominoes.

We show that such a removable tile can always be found if P is the 2-scaled copy of a non-degenerate polyomino. One of the core ideas of our method is to make P weakly

3-empty. If P is weakly 3-empty, we consider a partition of P into horizontal *slabs*. Based on this partition, we show that either a leaf of the dual graph of the partition can be removed, or a hole of P can be *cut open*, i.e., two adjacent tiles of P can be removed to reduce the number of holes by 1.

Definition 17 A *slice* of a shape P is the set of all tiles sharing the same y-coordinate. A *slab* is a maximal connected set of tiles within a slice.

Definition 18 Let P be a shape, and $\mathcal{S}_P$ its partition into slabs. For 2D shapes, or 3D shapes, we refer by $\mathcal{C}(\mathcal{S}_P)$ to the edge-contact graph, or to the face-contact graph of $\mathcal{S}_P$, respectively. That is, each slab is represented by a vertex, and two vertices are connected if and only if the union of both slabs is connected.

It is straightforward to see that any leaf of $\mathcal{C}(\mathcal{S}_P)$ can be removed if it lies in the outer face and the polyomino is weakly 3-empty.

Lemma 19 *Let P be a non-degenerate, weakly 3-empty polyomino. If $\mathcal{C}(\mathcal{S}_P)$ contains a leaf that lies in the outer face, then the corresponding slab S can be removed from P.*

Proof Without loss of generality, we assume that there is no neighbor below S and that $P \setminus t$ is connected, where t is the rightmost tile of S. Analogous arguments hold in case that t is the leftmost tile of a slab S.

Consider position of t and the position p which lies one step below and one step to the right of t. Because S is weakly 3-empty, p is free and it is the center of a 3×3 square that contains t in the top left corner. Because P is non-degenerate and by assumption the slab S has no neighbor to the bottom side, we can move t to p by moving it one step down and then one step right. Because P is weakly 3-empty, t can be removed. It is straightforward to see that $P \setminus t$ is still weakly 3-empty.

Because we can iteratively remove the leftmost or the rightmost tile of S without losing connectivity, we eventually remove S from P.   □

Theorem 20 *For every non-degenerate polyomino P, its 2-scaled copy P^2 is constructible.*

Proof The approach works in two phases. The first phase removes tiles from P^2 such that the remaining shape is weakly 3-empty. The second phase deconstructs the remaining shape slab by slab. Any holes that may be contained in the shape are cut open at appropriate times within the deconstruction sequence at suitable positions. If necessary, these phases are executed again one after the other.

Phase 1 works as follows, see Fig. 12: Consider the set F that contains all free positions that can participate in a pair with a free position outside the bounding box of P^2, such that these pairs fulfill the weakly 3-empty property. We will describe how tiles of P^2 can be removed such that any position lying in the outer face will belong to this set.

As long as there are adjacent free positions that are not in F, repeat the following: Let $p_1, p_2 \notin F$ be two adjacent free positions that both have a neighbor in F, say p'_1 and p'_2 respectively. Without loss of generality, let (p_1, p_2) be a vertical pair and let

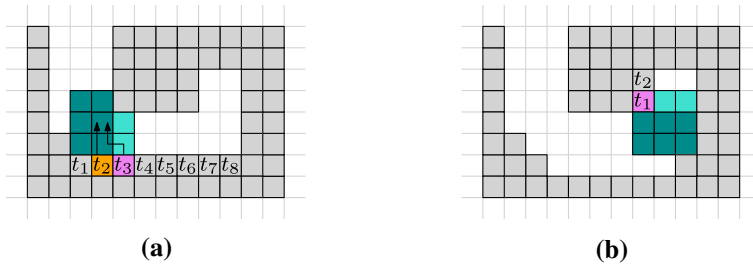**Fig. 12** (**a**) This figure visualizes the first case in the proof of Theorem 20. The horizontal line $L = (t_1, \ldots, t_8)$ with $t_b = t_3$. Cyan and teal positions must be free, teal positions correspond to the pair (p'_1, p'_2). The orange tile is removed first, followed the purple tile. (**b**) A visualization of the second case, i.e., when $t_b = t_1$. We can immediately remove t_1. Note that this figure is rotated by 90 degrees compared to the arguments given in the proof (Color figure online)

the neighbors be in the left direction. Then, to the top and to the bottom of (p'_1, p'_2) there are tiles $t_t, t_b \in P^2$ (or else $p_1, p_2 \in F$). Consider the maximal horizontal line L of m tiles $(t_1, \ldots, t_m)$ ordered from left to right with $t_b \in L$ that have a free position to the top. We make the following case distinction:

Case 1: $t_b \neq t_1$: Remove the tile t that is right of t_b by moving t two steps to the top. This position is the position to the left of p'_2, i.e., a position from F. Furthermore, this position has to be the center of a 3×3 square containing only t_t. Thus, t can be removed from P^2. After t is removed, t_b can be moved one step to the top, followed by a step right and a step up, reaching the exact same position.

Case 2: $t_b = t_1$: Remove t_b by moving the tile one step to the top, followed by a step to the right.

After removing t_b, we can proceed by iteratively removing L. Note that in some cases t_1 or t_m may not be removable, because they lie in a corner. However, the free positions adjacent to the top of these tiles will be contained in F, and thus, the corridor is wide enough.

The cases when p_1, p_2 have their neighbors from F in the direction to the right, or when they are a horizontal pair, are handled analogously. To maintain connectivity, we always choose the line L in a way that L is to the left, or to the down direction of (p'_1, p'_2).

For Phase 2 we can assume that the remaining polyomino P' is weakly 3-empty. Thus, we can apply Lemma 19 to remove slabs that correspond to leaves. If there is no such slab, then we either removed every single tile, or there is a slab S incident to the outer face with k neighbors in $\mathcal{C}(\mathcal{S}_{P'})$ on one side only, whose removal would produce at most $k - 1$ connected components. This slab must exist. To see this, consider the set of all slabs that have their neighbors in $\mathcal{C}(\mathcal{S}_{P'})$ on one side only, and consider a maximal path in $\mathcal{C}(\mathcal{S}_{P'})$ between these slabs. Then the slab S' is a slab with the desired property, or else we can extend the path.

Now, consider a slab S as defined above. We know that there are two slabs S_1 and S_2 adjacent to S that belong to the same connected component in $P' \setminus S$. Without loss of generality, let S_1 be to the left of S_2. The first two tiles t_1 and t_2 from S that have a

larger x-coordinate than the rightmost tile in S_1 can be removed. There cannot be any tile above or below t_1 and t_2, because we consider the 2-scaled copy of a polyomino, and thus, this cannot break any connectivity. Also, any slab connected to S to the left of t_1 remains connected to any slab that is connected to S to the right of t_2. This implies that $P' \setminus \{t_1, t_2\}$ is a connected polyomino.

However, $P' \setminus \{t_1, t_2\}$ may no longer be weakly 3-empty. By restarting Phase 1, further tiles can be removed. Because tiles are getting removed in each case, P^2 is eventually deconstructed. Thus, we conclude that for every initial polyomino P, its 2-scaled copy P^2 is constructible.   □

Because there are (even tree-shaped) indestructible polyominoes, this result is tight. If the polyomino is already 3-empty, we can skip the first phase. Whenever we have to cut open a hole, we remove three (instead of two) tiles, such that the property of being 3-empty is preserved. This results in the following corollary.

Corollary 21 *Every non-degenerate, 3-empty polyomino is constructible.*

In the three-dimensional setting we cannot simply widen narrow corridors when scaled by a factor of 2, e.g., imagine a tube with an inner diameter of 2. Therefore, the approach in 2D does not work in 3D. However, we show that a scaling factor of 3 suffices to guarantee constructibility.

Definition 22 Let S be a slab of a 3D shape, and S_1, S_2 be two slabs adjacent to S. S_1 is called *ring* if there are tiles of S_1 bounding a (3D) hole of P. S_1 and S_2 build a *loop* (or 2D hole), if both are in the same connected component of $P \setminus S$. If both cases do not apply, S_1 is called a *pillar*.

Using the same arguments as in the 2D case, we obtain the following lemma.

Lemma 23 *Let P be a 3D shape and $\mathcal{S}_P$ a decomposition of P into slabs. If $\mathcal{S}_P$ contains no slab of degree one, then there exists a slab accessible from the outer face with at most one pillar (outside a 3D hole).*

Proof Consider the set of all slabs adjacent to the outer face, that have their neighbors on one side only. In this set, there is a maximal path of these slabs using pillars that are adjacent to the outer face. Because P is finite, the start and end slab of this path can only have one pillar.   □

Theorem 24 *For every non-degenerate polycube P, its 3-scaled copy P^3 is constructible.*

Proof Similar to the 2D case, we proceed in two phases. In Phase 1, we successively remove slabs that are leaves in $\mathcal{C}(\mathcal{S}_P)$. Because we consider the 3-scaled copy of a shape, there is always sufficient space to remove all tiles from these slabs (see also 2D algorithm). If every slab has degree at least two, we proceed with Phase 2. Note that leaves can still exist, but these are not adjacent to the outer face and therefore not removable.

In Phase 2, we first search for a slab S that (i) is adjacent to the outer face, (ii) has neighbors on one side only, and (iii) is adjacent to a ring S_r. Consider a hole H that is
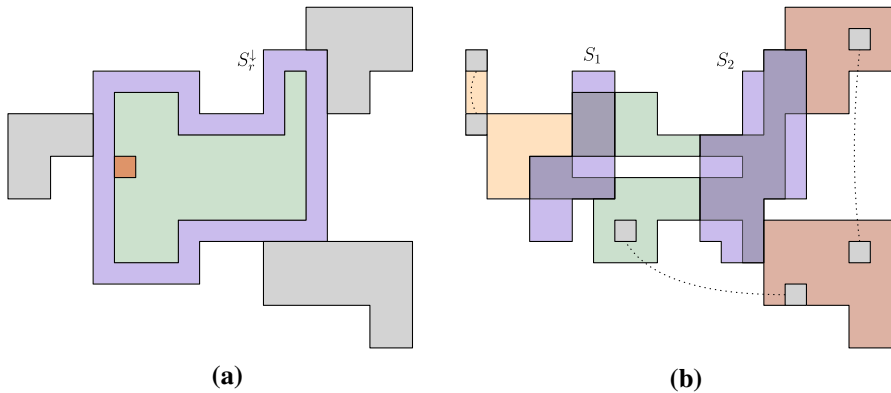
Fig. 13 (**a**) A slab S with tiles $S_r^{\downarrow}$ adjacent to a ring (shown in blue) and adjacent to a hole (green). We remove a 3×3 square from the green area (shown in red). Gray area is part of S but is not needed to consider here. (**b**) Blue area corresponds to slabs S_1 and S_2, dark blue area corresponds to tiles from S, that are below S_1 and S_2, i.e., S_1' and S_2'. The orange area corresponds to a connected component of Type 1, red areas to components of Type 2, and the green areas to components of Type 3. Gray squares indicate loops, when connected by a dotted curve (Color figure online)

bounded by tiles of S and S_r. Because S has neighbors on one side only, S covers one face of H completely. Let $S_r^{\downarrow}$ be the tiles from S that are adjacent to S_r. Then a 3×3 square from S that is adjacent to H and $S_r^{\downarrow}$ can be removed (see also Fig. 13(a)).

In the case that S does not exist, then, because of Lemma 23, there must be at least one slab S' with at least one adjacent loop. Let S_1, S_2 be two slabs adjacent to S' building a loop, and let S_1', S_2' be the set of tiles from S' adjacent to S_1 and S_2, respectively. Consider the set C of maximal connected components of $S \setminus \{S_1', S_2'\}$. The components of C can be divided into the following three types (see also Fig. 13(b)):

Type 1: Components that are adjacent only to tiles that belong to S_1'.
Type 2: Components that are adjacent only to tiles that belong to S_2'.
Type 3: Components that are adjacent only to tiles that belong to both S_1' and S_2'.

Any component that has no adjacent slab above/below itself can be removed. In the case that thereby all components of Type 3 have been removed, we proceed again with Phase 1. Otherwise, we make the following case distinction:

Case 1: Only components of Type 1 and Type 3 are still present. In this case, we remove S_2' and all tiles adjacent to S_2'. This keeps the remaining shape connected and non-degenerate, while also giving sufficient space to remove S_2 if it becomes a degree-one slab.
Case 2: Only components of Type 2 and Type 3 are still present. This is analogue to Case 1.
Case 3: Only components of Type 3 are present. This is analogue to Case 1.
Case 4: Components of all three types are still present. Without loss of generality, assume that the only pillar (if existing) is adjacent to a component of Type 1 or Type 3; otherwise we switch S_1 and S_2. Consider all pairs of slabs adjacent to S' that build a loop. Suppose there is a loop build by slabs $\bar{S}_1$ and $\bar{S}_2$ both
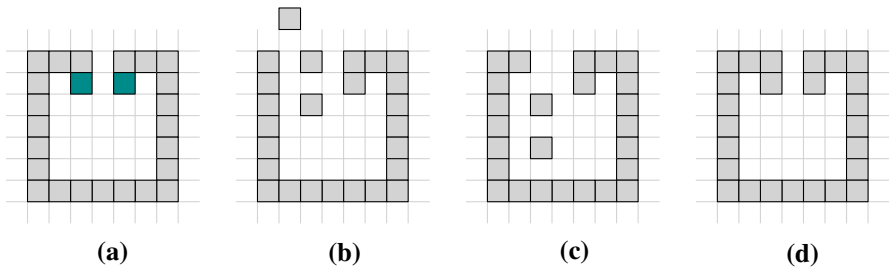
Fig. 14 The shape on the left is not constructible if only one tile at a time is allowed to be controlled. Because tiles stick together when they are on adjacent positions, no leaf (dark cyan tiles) can be removed, as well as no other tile without losing connectivity. If we instead are allowed to add multiple tiles simultaneously, the polyomino can be constructed as seen in (**b**) to (**d**) by two down tilts followed by three up tilts. The polyomino is also constructible if we are allowed to place two (dark cyan colored) seed tiles in advance. Note that in this case even adding one tile at a time is sufficient to construct the polyomino (Color figure online)

adjacent to components of Type 2 (see rightmost gray squares in the red area in Fig. 13(b)). Then, we restart Phase 2 with slab S', and the loop build by $\bar{S}_1$ and $\bar{S}_2$, such that the former slabs S_1 and S_2 are then adjacent to components of Type 1 or Type 3. This ensures that we do not cycle back to these two slabs.

We repeat this procedure until we have the following situation: (i) Every loop beginning in a component of Type 2 has its end in a component of Type 1 or Type 3, and (ii) the only pillar (if existing) is connected to a component of Type 1 or Type 3. With that, we can simply remove any component of Type 2 without losing connectivity, and we can continue with Case 1.

In each iteration of this phase, at least one loop is getting removed or one hole is cut open. Therefore, Phase 2 can only have finitely many iterations and thus eventually terminates. Combining both phases shows that the 3-scaled copy of every non-degenerate 3D shape is constructible.   □

Unlike as in the 2D case, we cannot show that this result is tight and conjecture that even the 2-scaled copy of every non-degenerate polycube is constructible.

6 Conclusion and Future Work

We provided a number of algorithmic results for assembling shapes by connecting particles one after the other to a fixed seed tile in a single step model. For future research several interesting problems remain open. We showed that constructibility can be decided for special classes of shapes, but we do not know how hard it is to decide whether an arbitrary polyomino is constructible or not in the considered model, i.e., what is the computational complexity of 2D-STAP? It seems that a similar construction as in the three-dimensional setting should work; however, the main difficulty is guaranteeing connectivity during a feasible deconstruction. We note that this is also an

open question in the full tilt model [11]. Another question is whether 3D-STAP remains NP-complete when restricted to the class of non-degenerate shapes.

In this paper we add one tile after the other to an assembly. If this assumption is relaxed, i.e., more than one tile at a time can be added to and controlled in the workspace, it is easy to see that more shapes are constructible, e.g., see Fig. 14. Is there a classification of shapes that can be built in this model? This also leads to the question: "Which shapes are constructible by using pre-assembled shapes (e.g., trominoes, tetrominoes, etc.)?" By taking this a step further, we could also ask for a staged approach similar to [44], where whole subassemblies can attach to each other. Another question arises by considering multiple seed tiles. What classes of shapes are constructible, if multiple seed tiles can be placed in advance, again see Fig. 14.

Is even the 2-scaled copy of every non-degenerate polycube constructible? Is there a similar example as in Fig. 10 for degenerate polycubes, i.e., is there a polycube such that no scaling factor guarantees constructibility in the case of degeneracy?

Our considerations are based on a tile with a single glue type on all sides so that tiles immediately stick together when they are on adjacent positions. It is a natural following question to ask what changes once we have different glue types.

Related version A preliminary extended abstract [32] of this work is published in the proceedings of the *17th Algorithms and Data Structures Symposium* (WADS 2021).

Additional material The 3D models of the gadgets used in the NP-hardness reduction can be accessed through https://gitlab.ibr.cs.tu-bs.de/alg/particle-based-assembly-extra

References

1. Agarwal, P.K., Aronov, B., Geft, T., et al.: On two-handed planar assembly partitioning with connectivity constraints. In: Symposium on Discrete Algorithms (SODA). SIAM, pp. 1740–1756. (2021). doi: https://doi.org/10.1137/1.9781611976465.105
2. Balanza-Martinez, J., Luchsinger, A., Caballero, D., et al.: Full tilt: Universal constructors for general shapes with uniform external forces. In: Symposium on Discrete Algorithms (SODA). SIAM, pp. 2689–2708. (2019). https://doi.org/10.1137/1.9781611975482.167
3. Balanza-Martinez, J., Gomez, T., Caballero, D., et al.: Hierarchical shape construction and complexity for slidable polyominoes under uniform external forces. In: Symposium on Discrete Algorithms (SODA). SIAM, pp. 2625–2641. (2020). doi: https://doi.org/10.1137/1.9781611975994.160

4. Becker, A.T., Demaine, E.D., Fekete, S.P., et al.: Reconfiguring massive particle swarms with limited, global control. In: Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics (ALGOSENSORS). Springer, pp. 51–66. (2013). doi: https://doi.org/10.1007/978-3-642-45346-5_5

5. Becker, A.T., Ou, Y., Kim, P.S.S., et al.: Feedback control of many magnetized: Tetrahymena pyriformis cells by exploiting phase inhomogeneity. In: International Conference on Intelligent Robots and Systems (IROS). IEEE, pp. 3317–3323. (2013). doi: https://doi.org/10.1109/IROS.2013.6696828

6. Becker, A.T., Demaine, E.D., Fekete, S.P., et al.: Particle computation: Designing worlds to control robot swarms with only global signals. In: International Conference on Robotics and Automation (ICRA). IEEE, pp. 6751–6756. (2014). doi: https://doi.org/10.1109/ICRA.2014.6907856

7. Becker, A.T., Felfoul, O., Dupont, P.E.: Simultaneously powering and controlling many actuators with a clinical MRI scanner. In: nternational Conference on Intelligent Robots and Systems (IROS). IEEE, pp. 2017–2023. (2014). https://doi.org/10.1109/IROS.2014.6942831

8. Becker, A.T., Felfoul, O., Dupont, P.E.: Toward tissue penetration by mri-powered millirobots using a self-assembled gauss gun. In: International Conference on Robotics and Automation (ICRA). IEEE, pp. 1184–1189. (2015). doi: https://doi.org/10.1109/ICRA.2015.7139341

9. Becker, A.T., Demaine, E.D., Fekete, S.P., et al.: Particle computation: Complexity, algorithms, and logic. Natural Comput. **18**(1), 181–201 (2019). https://doi.org/10.1007/s11047-017-9666-6

10. Becker, A.T., Fekete, S.P., Huang, L., et al.: Targeted drug delivery: Algorithmic methods for collecting a swarm of particles with uniform, external forces. In: International Conference on Robotics and Automation (ICRA). IEEE, pp. 2508–2514. (2020). doi: https://doi.org/10.1109/ICRA40945.2020.9196551

11. Becker, A.T., Fekete, S.P., Keldenich, P., et al.: Tilt assembly: Algorithms for micro-factories that build objects with uniform external forces. Algorithm. **82**(2), 165–187 (2020). https://doi.org/10.1007/s00453-018-0483-9

12. de Berg, M., Khosravi, A.: Optimal binary space partitions for segments in the plane. Inter. J. on Comput. Geometry and Appl. **22**(3), 187–206 (2012). https://doi.org/10.1142/S0218195912500045

13. Brown, I.D., Connolly, J.G., Kerkut, G.A.: Galvanotaxic response of tetrahymena vorax. Comparative Biochemistry and Physiology Part C: Comparative Pharmacology (1981). https://doi.org/10.1016/0306-4492(81)90140-4

14. Caballero, D., Cantu, A.A., Gomez, T., et al.: Relocating units in robot swarms with uniform gontrol signals is PSPACE-complete. In: Canadian Conference on Computational Geometry (CCCG) (2020)

15. Caballero, D., Cantu, A.A., Gomez, T., et al.: Building patterned shapes in robot swarms with uniform control signals. In: Canadian Conference on Computational Geometry (CCCG) (2020)

16. Caballero, D., Cantu, A.A., Gomez, T., et al.: Hardness of reconfiguring robot swarms with uniform external control in limited directions. J. of Inform. Process. **28**, 782–790 (2020). https://doi.org/10.2197/ipsjjip.28.782

17. Caballero, D., Cantu, A.A., Gomez, T., et al.: Fast reconfiguration of robot swarms with uniform control signals. Natural Comput. **20**, 659–669 (2021). https://doi.org/10.1007/s11047-021-09864-0

18. Cannon, S., Demaine, E.D., Demaine, M.L., et al.: Two hands are better than one (up to constant factors): Self-assembly in the 2ham vs. atam. In: International Symposium on Theoretical Aspects of Computer Science (STACS), pp. 172–184. (2013). doi: https://doi.org/10.4230/LIPIcs.STACS.2013.172

19. Chalk, C.T., Martinez, E., Schweller, R.T., et al.: Optimal staged self-assembly of general shapes. Algorithm. **80**(4), 1383–1409 (2018). https://doi.org/10.1007/s00453-017-0318-0

20. Chalk, C.T., Martinez, E., Schweller, R.T., et al.: Optimal staged self-assembly of linear assemblies. Natural Comput. **18**(3), 527–548 (2019). https://doi.org/10.1007/s11047-019-09740-y

21. Chanu, A., Felfoul, O., Beaudoin, G., et al.: Adapting the clinical mri software environment for real-time navigation of an endovascular untethered ferromagnetic bead for future endovascular interventions. Magnetic Resonance in Medicine: An Official J. of the Int. Soc. for Magnetic Resonance in Medicine **59**(6), 1287–1297 (2008). https://doi.org/10.1002/mrm.21638

22. Cheang, U.K., Roy, D., Lee, J.H., et al.: Fabrication and magnetic control of bacteria-inspired robotic microswimmers. Appl. Phys. Letters **97**(21), 213–704 (2010). https://doi.org/10.1063/1.3518982

23. Chen, H., Doty, D.: Parallelism and time in hierarchical self-assembly. SIAM J. of Comput. **46**(2), 661–709 (2017). https://doi.org/10.1137/151004161

24. Demaine, E.D., Demaine, M.L., Fekete, S.P., et al.: Staged self-assembly: nanomanufacture of arbitrary shapes with O(1) glues. Natural Comput. **7**(3), 347–370 (2008). https://doi.org/10.1007/s11047-008-9073-0

25. Demaine, E.D., Demaine, M.L., Fekete, S.P., et al.: One tile to rule them all: Simulating any tile assembly system with a single universal tile. In: International Colloquium on Automata, Languages, and Programming (ICALP), pp. 368–379. (2014). doi: https://doi.org/10.1007/978-3-662-43948-7_31

26. Demaine, E.D., Fekete, S.P., Scheffer, C., et al.: New geometric algorithms for fully connected staged self-assembly. Theor. Comput. Sci. **671**, 4–18 (2017). https://doi.org/10.1016/j.tcs.2016.11.020

27. Doty, D.: Theory of algorithmic self-assembly. Commun. of the ACM **55**(12), 78–88 (2012). https://doi.org/10.1145/2380656.2380675

28. Dreyfus, R., Baudry, J., Roper, M.L., et al.: Microscopic artificial swimmers. Nature **437**, 862–865 (2005). https://doi.org/10.1038/nature04090

29. Fu, B., Patitz, M.J., Schweller, R.T., et al.: Self-assembly with geometric tiles. In: International Colloquium on Automata, Languages, and Programming (ICALP), pp. 714–725. (2012). doi: https://doi.org/10.1007/978-3-642-31594-7_60

30. Ghosh, A., Fischer, P.: Controlled propulsion of artificial magnetic nanostructured propellers. Nano letters **9**(6), 2243–2245 (2009). https://doi.org/10.1021/nl900186w

31. Keldenich, P., Manzoor, S., Huang, L., et al.: On designing 2d discrete workspacesto sort or classify polyominoes. In: International Conference on Intelligent Robots and Systems (IROS), pp. 1–9. (2018). doi: https://doi.org/10.1109/IROS.2018.8594150

32. Keller, J., Rieck, C., Scheffer, C., et al.: Particle-based assembly using precise global control. In: Algorithms and Data Structures (WADS). Springer, pp. 513–527. (2021). doi: https://doi.org/10.1007/978-3-030-83508-8_37

33. Kim, P.S.S., Becker, A.T., Ou, Y., et al.: Swarm control of cell-based microrobots using a single global magnetic field. In: International Conference on Ubiquitous Robots and Ambient Intelligence (URAI). IEEE, pp. 21–26. (2013). doi: https://doi.org/10.1109/URAI.2013.6677461

34. Kim, P.S.S., Becker, A.T., Ou, Y., et al.: Imparting magnetic dipole heterogeneity to internalized iron oxide nanoparticles for microorganism swarm control. Journal of Nanoparticle Research **17**(3), (2015). doi: https://doi.org/10.1007/s11051-014-2746-y

35. Mahadev, A.V., Krupke, D., Reinhardt, J., et al.: Collecting a swarm in a grid environment using shared, global inputs. In: International Conference on Automation Science and Engineering (CASE). IEEE, pp. 1231–1236. (2016). doi: https://doi.org/10.1109/COASE.2016.7743547

36. Manzoor, S., Sheckman, S., Lonsford, J., et al.: Parallel self-assembly of polyominoes under uniform control inputs. IEEE Robotics Automation Letters **2**(4), 2040–2047 (2017). https://doi.org/10.1109/LRA.2017.2715402

37. Martel, S., Felfoul, O., Mathieu, J.B., et al.: Mri-based medical nanorobotic platform for the control of magnetic nanoparticles and flagellated bacteria for target interventions in human capillaries. The Intern. J. of Robotics Research **28**(9), 1169–1182 (2009). https://doi.org/10.1177/0278364908104855

38. Patitz, M.J.: An introduction to tile-based self-assembly and a survey of recent results. Natural Comput. **13**(2), 195–224 (2014). https://doi.org/10.1007/s11047-013-9379-4

39. Peyer, K.E., Tottori, S., Qiu, F., et al.: Magnetic helical micromachines. Chemistry-A. European J. **19**(1), 28–38 (2013). https://doi.org/10.1002/chem.201203364

40. Peyer, K.E., Zhang, L., Nelson, B.J.: Bio-inspired magnetic swimming microrobots for biomedical applications. Nanoscale **5**(4), 1259–1272 (2013). https://doi.org/10.1039/c2nr32554c

41. Rothemund ,P.W.K.: Theory and experiments in algorithmic self-assembly. PhD thesis, University of Southern California (2001)

42. Rothemund, P.W.K.: Design of DNA origami. In: International Conference on Computer-Aided Design (ICCAD). IEEE Computer Society, pp. 471–478. (2005). https://doi.org/10.1109/ICCAD.2005.1560114

43. Rothemund, P.W.K., Winfree, E.: The program-size complexity of self-assembled squares. In: Symposium on Theory of Computing (STOC). ACM, pp. 459–468. (2000). https://doi.org/10.1145/335305.335358

44. Schmidt, A., Manzoor, S., Huang, L., et al.: Efficient parallel self-assembly under uniform control inputs. IEEE Robotics Automation Letters **3**(4), 3521–3528 (2018). https://doi.org/10.1109/LRA.2018.2853758

45. Schulman, R., Winfree, E.: Synthesis of crystals with a programmable kinetic barrier to nucleation. Proc. of the National Academy of Sci. **104**(39), 15,236-15,241 (2007). https://doi.org/10.1073/pnas.0701467104

46. Schweller, R.T., Winslow, A., Wylie, T.: Nearly constant tile complexity for any shape in two-handed tile assembly. Algorithm **81**(8), 3114–3135 (2019). https://doi.org/10.1007/s00453-019-00573-w

47. Shad, H.M., Morris-Wright, R., Demaine, E.D., et al.: Particle computation: Device fan-out and binary memory. In: International Conference on Robotics and Automation (ICRA). IEEE, pp. 5384–5389. (2015). doi: https://doi.org/10.1109/ICRA.2015.7139951

48. Soloveichik, D., Winfree, E.: Complexity of self-assembled shapes. SIAM J. Comput. **36**(6), 1544–1569 (2007). https://doi.org/10.1137/S0097539704446712

49. Steager, E., Kim, C.B., Patel, J., et al.: Control of microfabricated structures powered by flagellated bacteria using phototaxis. Appl. Phys. Letters **90**(26), 263–901 (2007). https://doi.org/10.1063/1.2752721

50. Vartholomeos, P., Akhavan-Sharif, M.R., Dupont, P.E.: Motion planning for multiple millimeter-scale magnetic capsules in a fluid environment. In: International Conference on Robotics and Automation (ICRA). IEEE, pp. 1927–1932. (2012). doi: https://doi.org/10.1109/ICRA.2012.6225330

51. Vartholomeos, P., Bergeles, C., Qin, L., et al.: An mri-powered and controlled actuator technology for tetherless robotic interventions. Intern. J. of Robotics Research **32**(13), 1536–1552 (2013). https://doi.org/10.1177/0278364913500362

52. Wang, H.: Proving theorems by pattern recognition-II. Bell Syst. Technical J. **40**(1), 1–41 (1961). https://doi.org/10.1002/j.1538-7305.1961.tb03975.x

53. Wang, H.: Dominoes and the AEA case of the decision problem. In: Computation, Logic, Philosophy. pp. 218–245. Springer (1990)

54. Weibel, D.B., Garstecki, P., Ryan, D., et al.: Microoxen: Microorganisms to move microscale loads. National Academy of Sci. **102**(34), 11,963-11,967 (2005). https://doi.org/10.1073/pnas.0505481102

55. Winfree, E.: Algorithmic self-assembly of DNA. PhD thesis, California Institute of Technology (1998)

56. Winfree, E., Liu, F., Wenzler, L.A., et al.: Design and self-assembly of two-dimensional dna crystals. Nature **394**(6693), 539–544 (1998). https://doi.org/10.1038/28998

57. Woods, D.: Intrinsic universality and the computational power of self-assembly. Philosophical Trans. of the Royal Society A: Math., Phys. and Eng. Sci. **373**(2046), 20140,214 (2015). https://doi.org/10.1098/rsta.2014.0214

58. Zhang, L., Abbott, J.J., Dong, L., et al.: Artificial bacterial flagella: Fabrication and magnetic control. Appl. Phys. Letters **94**(6), 064,107 (2009). https://doi.org/10.1063/1.3079655

59. Zhang, L., Peyer, K.E., Nelson, B.J.: Artificial bacterial flagella for micromanipulation. Lab Chip **10**(17), 2203–2215 (2010). https://doi.org/10.1039/C004450B