Efficient Parallel Self-Assembly Under Uniform Control Inputs

Arne Schmidt ⓘ, Sheryl Manzoor ⓘ, Li Huang ⓘ, Aaron T. Becker ⓘ, and Sándor P. Fekete ⓘ

Abstract—We prove that by successively combining subassemblies, we can achieve *sublinear* construction times for "staged" assembly of microscale objects from a large number of tiny particles, for vast classes of shapes; this is a significant advance in the context of programmable matter and self-assembly for building high-yield microfactories. The underlying model has particles moving under the influence of uniform external forces until they hit an obstacle; particles bond when forced together with a compatible particle. Previous work considered sequential composition of objects, resulting in construction time that is *linear* in the number N of particles, which is inefficient for large N. Our progress implies critical speedup for constructible shapes; for convex polyominoes, even a *constant* construction time is possible. We also show that our construction process can be used for pipelining, resulting in an *amortized constant* production time.

Index Terms—Computational geometry, underactuated robots, additive manufacturing.

I. INTRODUCTION

THE new field of programmable matter gives rise to a wide range of algorithmic questions of geometric flavor. One of the tasks is designing and running efficient production processes for tiny objects with given shape, without being able to individually handle the potentially huge number of particles from which it is composed, e.g., building polyominoes from their tiles without the help of tools.

In this letter we use particles that can be controlled by a uniform external force, causing all particles to move in a given direction until they hit an obstacle or another blocked particle, as shown in Fig. 1.

Recent experimental work by Manzoor *et al.* [11] showed this is practical for simple "sticky" particles, enabling assembly by
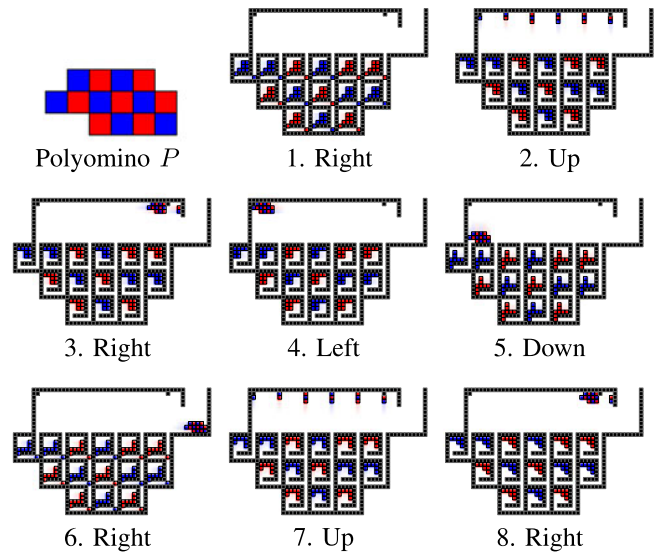
Fig. 1. Convex polyominoes can be assembled in six movement steps. A copy of the polyomino P is released every five steps after the first copy. See video attachment for animation: https://youtu.be/_R_puO0smPs.

sequentially attaching particles emanating from different depots within the workspace or supply channels from the outside to the existing subassembly, as shown in Fig. 1.

The algorithmic challenge is to design the surrounding "maze" environment and movement sequence to produce a desired shape.

A recent paper by Becker *et al.* [3] showed that the decision problem of whether a simple polyomino can be built or not is solvable in polynomial time. However, this relies on sequential construction in which one particle at a time is added, resulting in a *linear* number of assembly steps, i.e., a time that grows proportional to the number N of particles, which is inefficient for large N. In this letter we provide substantial progress by developing methods that can achieve *sublinear* and in some cases even *constant* construction times. Our approaches are based on hierarchical, "staged" processes, in which we allow multi-tile subassemblies to combine at each construction step.

A. Contribution

We provide a number of contributions to achieving sublinear construction times for polyomino shapes consisting of N pixels ("tiles"), which is critical for the efficient assembly of large objects. Many of these results are the outcome of decomposing the shape into simpler pieces; as a consequence, we can

describe the construction time in geometric parameters that may be considerably smaller than N.

- We show that we can decide if a given polyomino P can be recursively constructed from simple subpieces that are glued together along simple straight cuts ("2-cuts") in polynomial time. The resulting production time depends on the number $r(P)$ of *locally reflex* tiles of P, which is bounded by N, but may be much smaller.
- We show that building a convex polyomino takes $O(1)$ steps.
- For a monotone polyomino P, we need $O(\log \mathrm{d}(P))$ steps, where $\mathrm{d}(P) \leq N$ is the number of cuts needed to decompose P into convex subpolyominoes.
- For polyominoes with convex holes, we show that $O(r)$ steps suffice to build the polyomino.
- All methods we describe can be pipelined resulting in an amortized constant construction time.

We also elaborate the running time for efficiently *computing* aspects of the decomposition, as follows. Finding cuts for a decomposition needs $O(N)$ time for monotone polyominoes. Simple polyominoes require $O(N + r^2 \log r)$ time to find a straight cut and $O(r^2 N \log N)$ time to find an arbitrary cut. Allowing convex holes increases the time to $O(N + r^3 \log r)$ and $O(r^3 N \log N)$, respectively.

For all these constructions, we show that $N \cdot (\mathcal{C}_P + \sqrt{D})$ obstacles suffice to construct D copies of an N-tile polyomino that requires $\mathcal{C}_P$ steps to build.

B. Related Work

In recent years, the problem of assembling a polyomino has been studied intensively using various theoretical models. Winfree [14] introduced the *abstract tile self-assembly model* in which tiles with glues on their side can attach to each other if their glue type matches. Then, starting with a *seed-tile*, the tiles continuously attach to the partial assembly. If no further tile can attach, the process stops. Several years later, Cannon *et al.* [5] introduced the *2-handed tile self-assembling model (2HAM)* in which sub-assemblies can attach to each other provided that the sum of glue strengths is at least a threshold τ. Chen and Doty [7] introduced a similar model: the *hierarchical tile self-assembling model*. In 2008, Demaine *et al.* [8] introduced the *staged tile self-assembly model* which is based on the 2HAM. Here, sub-assemblies grow in various bins which can then poured together to gain new assemblies. This model was then further analyzed by Demaine *et al.* [9] and Chalk *et al.* [6]. An interesting aspect in all models is that the third dimension can be used to reach specific positions within partial assemblies. In our paper however, the challenge is to use two dimensions, i.e., an assembly can only bond to another polyomino if the bonding site is completely visible.

All these models have in common that particles, e.g., DNA-strands, self-assemble to bigger structures. In this letter, however, the particles can only move by global controls and have one glue type on all four sides. This concept has been studied in practice using biological cells controlled by magnetic fields, see [10]. In addition, see [1]. Recent work by Zhang *et al.* [15] shows there exists a workspace a constant factor larger than the number of agents that enables complete rearrangement for a rectangle of agents.

A more related paper is the work by Manzoor *et al.* [11]. They assemble polyominoes in a pipelined fashion using global control, i.e., by completing a polyomino after each small control sequence the amortized construction time of a polyomino is constant. To find a construction sequence building the polyomino only heuristics are used. Becker *et al.* [3] show that it is possible to decide in polynomial time if a hole-free polyomino can be constructed. However, both papers consider adding one tile at a time. In this letter, we allow combining partial assemblies at each step. We are also able to pipeline this process to achieve an amortized constant production time.

The complexity of controlling robots using a global control has been studied. Becker *et al.* [2] show that it is NP-hard to decide if an initial configuration of a robot swarm in a given environment can be transformed into another configuration by only using global control but becomes more tractable if it is allowed to *design* the environment. Finding an optimal control sequence is even harder. Related work for reconfiguration of robots with local movement control include work by Walter *et al.* [13], Vassilvitskii *et al.* [12], and Butler *et al.* [4].

II. PRELIMINARIES

Workspace: A *workspace* $\mathcal{W}$ is a planar grid filled with unit-square particles and fixed unit square blocks (*obstacles*). Each cell of the workspace contains either a particle, an obstacle, or the cell is *free*.

Movement step: A *movement step* is one of the four directions *up*, *right*, *down*, *left*. One movement step forces every tile or assembly to move to the specified direction until the tile/assembly is blocked by an obstacle.

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. A polyomino is called *hole-free* or *simple* if and only if the grid graph induced by $\mathbb{Z}^2 \setminus P$ is connected. A polyomino P is *column convex* (*row convex*, resp.) if the intersection of any vertical (horizontal, resp.) line and P is connected, i.e., the polyomino is x-monotone (y-monotone, resp.). Furthermore, a polyomino P is called *(orthogonal) convex* if P is column and row convex.

Tiles: A tile t is an unit-square of a polyomino and also represent particles in the workspace. There are two kinds of tiles: blue and red tiles. Two tiles stick together if their color differs.

Constructibility: A polyomino P is *constructible* if there exists a workspace $\mathcal{W}$ and a sequence σ of movement steps that produce P.

Cuts: A *cut* is an orthogonal curve moving between points of $\mathbb{Z}^2$. If any intersection of a cut with the polyomino P has no turn, the cut is called *straight*. A *p-cut* is a cut that splits a polyomino P into p subpolyominoes. Furthermore, a cut is called *valid* if all induced subpolyominoes can be pulled apart into opposite directions without blocking each other. A polyomino P is called *(straight) 2-cuttable* if there is a sequence of valid (straight) 2-cuts that subdivide P into monotone subpolyomi-
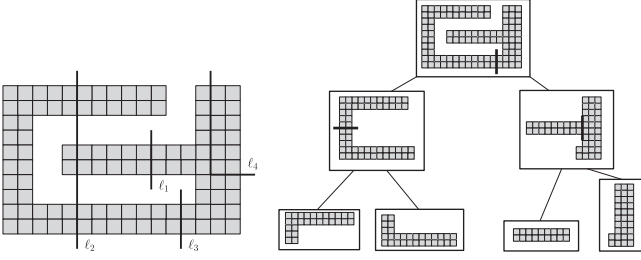
Fig. 2. Left: (Counter-)Examples for straight 2-cuts: ℓ_1 is not a 2-cut because we cannot move the left component to the right or left without getting blocked by the other component. ℓ_2 is not a 2-cut because we get more than two components. ℓ_3 is a 2-cut because we get two components which can be pulled apart. ℓ_4 is not a straight cut. Right: Decomposition tree with straight 2-cuts where the leaves are convex polyominoes.
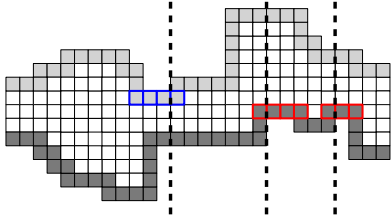

Fig. 3. A polyomino P. Light grey tiles define P_{up}, dark grey tiles define P_{lo}. Blue framed rows are minima in P_{up}, red framed rows are maxima in P_{lo}. Decomposition number $\mathtt{d}(P) = 3$, because three vertical lines suffice and three line are necessary because every line hits a maxima/minima.

noes. If the subpolyominoes can be pulled apart in horizontal (vertical) directions, we call the cut *vertical (horizontal)* An example for 2-cuts can be seen in Fig. 2. In the following we only consider 2-cuts for non-convex polyominoes.

III. MONOTONE ASSEMBLIES

This section focuses on convex and monotone polyominoes.

Lemma 1: Any convex polyomino P can be assembled in six movement steps.

Proof: The idea of this proof is simple: Subdivide P into vertical lines of width one, build the lines in two steps (see Fig. 1.1 and 1.2), and connect these lines with a right and left movement (see Fig. 1.3 and 1.4). With two more movements we can flush P out of the labyrinth (see Fig. 1.5 and 1.6).

Assembling a column: To construct a column of length n, we build n containers, each below the previous. Each container releases a new tile after each left, down, right movement combination. After the right movement all n tiles move to a wall and then have the same x-coordinate. With an up movement all n tiles stick to a column once the first tile hits the top wall.

Assembling the polyomino: Assume we have built each column of the polyomino in parallel. With obstacles we can stop each column at the appropriate respective heights. A right movement combines all columns left of the column with the maximum height, and a left movement completes the assembly of P. To remove the polyomino from the assembly area we use a down, right movement. Note that the last three movements are left, down, and right, by which we start the next copy of P.

Without further precautions, a polyomino could get stuck in narrow corridors. This problem can be avoided with a simple case analysis. First, observe that the leftmost of the topmost tiles of the polyomino is blocked by an obstacle. Let t be this tile and let x_t be the corresponding x-coordinate. Also, let s be a tile stuck in a corridor having x-coordinate x_s. Only two cases can occur. (a) $x_s < x_t$: We place an additional obstacle directly where t was blocked. This forces the polyomino to stop one position earlier. (b) $x_s > x_t$: We shift every obstacle with x-coordinate higher than the corridor one unit to the right and we add an additional obstacle at the corridor end. The polyomino is then stopped by this obstacle. □

Definition 1: Let P be an x-monotone polyomino. The decomposition number $\mathtt{d}(P)$ is the minimum number of vertical cuts required to obtain subpolyominoes that are all convex.

The *upper envelope* $P_{\text{up}} \subset P$ consists of (1) all tiles T on the boundary that have no tiles above, and (2) tiles connecting T along the boundary. Analogously define the *lower envelope* $P_{\text{lo}} \subset P$.

We call a straight row $M = \{m_1, \ldots, m_k\} \subset P_{\text{up}}$ a *minimum* of P_{up} if there are two tiles t_1 and t_2, for which t_1 is connected to the top side of m_1 and t_2 is connected to the top side of m_k. Analogously define *maximum* for P_{lo}.

To construct an x-monotone polyomino make vertical cuts through the maxima/minima of P_{up} and P_{lo}, respectively. There are at most $\mathtt{d}(P)$ many cuts. We now can choose a cut, such that on both subpolyominoes P' and P'' the decomposition number $\mathtt{d}(P') \leq \mathtt{d}(P'') \leq \frac{1}{2}\mathtt{d}(P)$; this can be done with a median search. Repeating this procedure on each resulting subpolyomino yields a decomposition tree with depth $\log \mathtt{d}(P)$ whose leafs are convex polyominoes.

Lemma 2: Let P be a polyomino. For each minimum and maximum M there must be a vertical cut ℓ going through M in order to decompose P into convex subpolyominoes.

Proof: Suppose we do not need such line ℓ. Let P' be a subpolyomino having a minimum M', through which no cut is made. Consider the two tiles t_1 and t_2 as defined above. Both t_1 and t_2 must be in the same subpolyomino (because there is no cut through M'). Then, a horizontal line through t_1 and t_2 enters P' twice and therefore, P' cannot be an convex polyomino. □

Lemma 3: Let P be an x-monotone polyomino. The decomposition number $\mathtt{d}(P)$ and the corresponding cuts can be computed in $O(N)$ time.

Proof: Finding the minima and maxima of P_{up} and P_{lo}, respectively, can be found in $O(N)$ time by sweeping from the left boundary to the right boundary. Having the minima M_{up} and maxima M_{lo}, both in sorted order from left to right, we repeat the following procedure:

- Let $M_0 \in M_{\text{up}}$ and $M_0' \in M_{\text{lo}}$ be the leftmost minima/maxima, resp.
- If the projection of M_0 and M_0' to the x-axis overlaps with at least two tiles, then output a vertical line going through M_0 and M_0', and remove both from M_{up} and M_{lo}, resp.
- If this is not the case, output a vertical line going through the minima/maxima that ends first, and remove this minima/maxima from M_{up} or M_{lo}, resp.

This procedure costs $O(\mathtt{d}(P))$ time. In total, this is $O(N)$ time. The correctness follows from Lemma 2. □

Fig. 4.   Assembling two subpolyominoes P_1 and P_2, where the topmost tile of P_1 lies above the topmost tile of P_2. These are the same movements as seen in Fig. 1 for convex polyominoes. Thus, we can combine two subpolyominoes while constructing the next convex subpolyomino. (a) Up. (b) Right. (c) Left. (d) Down. (e) Right. (f) Up.

Fig. 6.   A complete example constructing P with $\mathtt{d}(P) = 3$. State shown is after an up-movement and its previous state in translucent colors. Top-left box: A polyomino P with locally convex tiles (red), locally reflex tiles (blue), and tiles that are both locally convex and locally reflex (orange, striped).
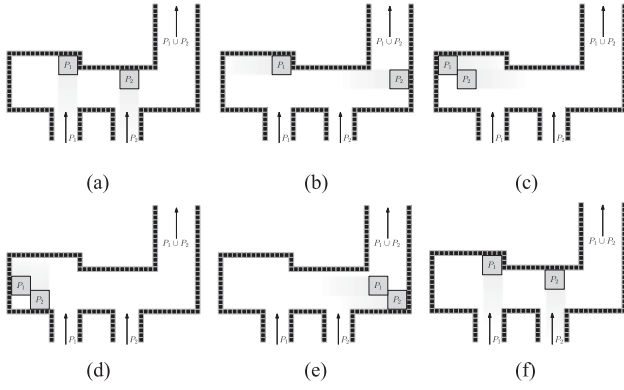

Fig. 5.   Assembling two subpolyominoes P_1 and P_2, where the topmost tile of P_2 lies above the topmost tile of P_1. These are the same movements as seen in Fig. 1 for convex polyominoes. Thus, we can combine two subpolyominoes while constructing the next convex subpolyomino. (a) Up. (b) Right. (c) Left. (d) Down. (e) Right. (f) Up.
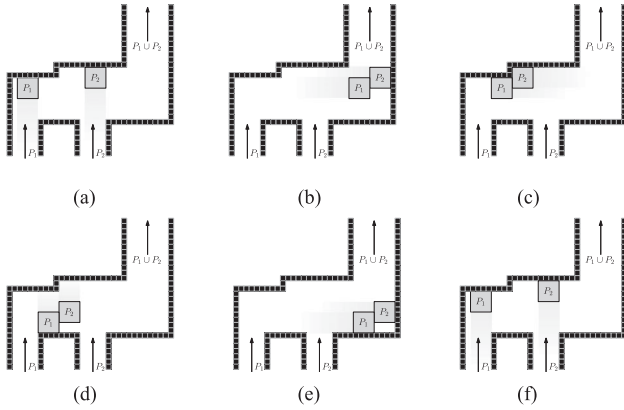
and polyominoes with convex holes. We end this section by showing how much space is needed for the workspace in which we can assemble the polyominoes.

A. Simple Polyominoes

To prove if a simple polyomino can be constructed we look at the converse process: a decomposition. As defined in the preliminary section we use 2-cuts to decompose a polyomino. If the polyomino cannot be decomposed by 2-cuts then the polyomino cannot be constructed by successively putting two subpolyominoes together. We show with the next lemma that we can greedily pick any valid straight 2-cut.

Lemma 4: Any valid straight 2-cut preserves decomposability.

Proof: Consider a straight 2-cut ℓ and a sequence $\sigma = (\ell_1, \ldots, \ell_m)$ of cuts, decomposing P into single tiles. Assume ℓ is part of the cut sequence but not the first cut in σ. Then, there is a 2-cut ℓ' being made directly before ℓ in a polyomino P^* induced by cuts before ℓ'. We can now swap ℓ' and ℓ preserving their property of being 2-cuts: for ℓ we assume it is a 2-cut in P, which is also true in any subpolyomino induced by 2-cuts; the same holds for ℓ', it is a 2-cut in P^* and thus, also in any subpolyomino induced by 2-cuts. After swapping both cuts we have the same decomposition yielding a valid decomposition of P. We can now repeat this procedure until ℓ is the first cut in P.

However, ℓ may not be in the cut sequence σ. We now show that we can use ℓ as a cut by exchanging cuts. Let ℓ_k be the last cut intersecting ℓ. This cut separates two cuts ℓ' and ℓ'' which lie on ℓ. Because ℓ is a 2-cut, also $\ell' \cup \ell''$ must be a 2-cut in the polyomino where we use cut ℓ_k. Therefore, we can first use the cut $\ell' \cup \ell''$ and then the two cuts ℓ'_k and ℓ''_k induced by the intersection of ℓ and ℓ_k. By repeating this procedure, we get ℓ as part of the cut sequence σ.   □

Definition 2: A tile t of a polyomino P is said to be *locally convex* if there exists a 2×2 square solely containing t. If the square only contains t and its two neighbors, then we call t *locally reflex*. Note that a tile can be locally convex and locally reflex at the same time (see Box in Fig. 6).

Lemma 5: Any non-convex, straight 2-cuttable polyomino P can be decomposed into convex subpolyominoes by only using straight 2-cuts cutting along a locally reflex tile.

Theorem 1: Any x-monotone polyomino P with decomposition number $\mathtt{d}(P) > 0$ can be assembled in $O(\lceil \log(1 + \mathtt{d}(P)) \rceil)$ unit steps. Furthermore, this process can be pipelined yielding a construction time of amortized $O(1)$ unit steps.

Proof: As a first step we search for the vertical cuts as described above. Having this subdivision into convex subpolyominoes, we can use Lemma 1 to create all subpolyominoes in parallel. We now can use the combining gadget seen in Figs. 4 and 5 to combine two adjacent subpolyominoes in each cycle. Thus, for each cycle the number of subpolyominoes decreases by a factor of two and we have at most $\lceil \log(1 + \mathtt{d}(P)) \rceil$ cycles to combine all subpolyominoes to obtain P.

As already described in Lemma 1, we start a new copy after every cycle. Thus, to create D copies of P we need $O(\lceil \log(1 + \mathtt{d}(P)) \rceil + D)$ cycles. This is an amortized constant time per copy if we create $\Omega(\log \mathtt{d}(P))$ copies. Note that $\mathtt{d}(P)$ is in $\Omega(1)$ and $O(N)$.   □

IV. Assembling Non-Monotone Shapes

In this section we show how to decide constructibility for special classes of polyominoes, namely simple polyominoes
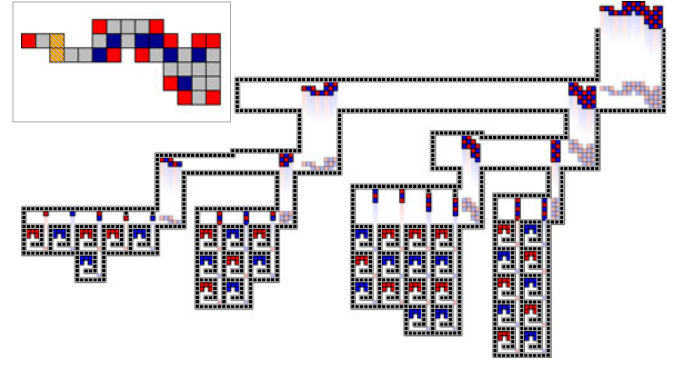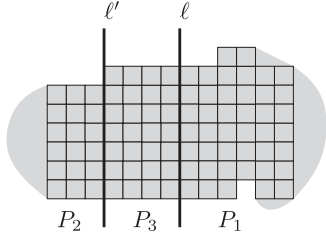
Fig. 7. The original cut ℓ and its shifted copy ℓ', which together split the polyomino into three parts P_1, P_2, P_3.
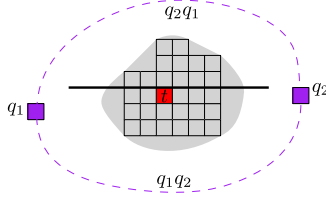

Fig. 8. A not locally convex tile t (red) in P_1 (gray area) blocked by q_1 and q_2 (purple). If the path $q_2 q_1$ exists, there is at least one blocked locally convex tile above the black bold line. If $q_1 q_2$ exists, we proceed analogously.

Proof: W.l.o.g., consider a vertical straight 2-cut ℓ that may not cut along a locally reflex tile. Then we can move a cut ℓ' to the left starting at ℓ until we reach a locally reflex tile t such that the cut goes through the corner of t that lies on the boundary of P (if we cannot reach a locally reflex tile we move ℓ' to the right). We obtain three subpolyominoes: P_1 to the right of ℓ, P_2 to the left of ℓ', and P_3 between ℓ' and ℓ (see Fig. 7).

Assume ℓ' is not a valid 2-cut, i.e., a tile is blocked in P_2 or P_3. (If there is a blocked tile in P_1, then also ℓ would not be a 2-cut.) Consider the first case, where P_2 has a blocked tile t. Then, t has an y-coordinate which is at most as high as the highest tile in P_3 plus 1 and at least as high as the lowest tile in P_3 minus one (or else both blocking tiles must be in P_1 and thus, ℓ would be no 2-cut). Let $q_1 \in P_3$ to the right of t and $q_2 \in P_1 \cup P_3$ to the left of t be the two tiles blocking t. By replacing q_1 with its right neighbor we still have two tiles blocking t. Because ℓ is a 2-cut we can repeat this procedure until $q_1 \in P_1$. We can repeat the procedure for q_2 if $q_2 \in P_3$. Thus, both blocking tiles are in P_1 and ℓ cannot be a 2-cut.

For the second case the blocked tile t lies in P_3. Then, also the right neighbor t' of t is blocked. This is also true for t'. Therefore, we can go to the right until we reach P_1 and thus, there is a tile in P_1 which is blocked. This means, also ℓ cannot be a valid 2-cut, which is a contradiction to ℓ being a valid 2-cut.

As each cut ℓ' reduces the number of locally reflex tiles by at least one, the remaining polyominoes will be convex after a limited number of cuts. □

Lemma 6: It is sufficient to consider locally convex tiles for checking if a cut ℓ is a valid straight 2-cut.

Proof: Assume w.l.o.g. ℓ is a vertical cut splitting the polyomino in two subpolyominoes P_1 and P_2. W.l.o.g., consider a not locally convex tile $t \in P_1$ blocked by two tiles $q_1, q_2 \in P_2$. Because ℓ is a 2-cut and P is simple, there must be a path from q_1 to q_2 within P_2. This path must go around P_1 either above or beneath t (see Fig. 8).

In case the path moves above t, consider a horizontal cut directly above t (see Fig. 8). This cut splits P_1 into components. In each component there are at least four locally convex tiles from which at most two became locally convex through the cut. Thus, two of these locally convex tiles were also locally convex in P_1. It is easy to see in the figure that both locally convex tiles are also blocked by tiles on the path from q_1 to q_2.

In the second case we proceed analogously with the difference that we use a horizontal cut directly below t. We conclude that in any case there is a locally convex tile in P_1 that is being blocked if there is a blocked, not locally convex tile. Note that the other direction may not be true. □

Lemma 7: Checking if a 2-cut ℓ is valid can be done in $O(N + r \log r)$ time, where r is the number of locally reflex tiles.

Proof: W.l.o.g. assume ℓ to be a vertical straight cut and also assume that we are checking blue tiles only. As a first step we scan through the polyomino and search for all tiles that represent a corner, i.e., the tile is locally convex or locally reflex. Additionally, we can store the neighbor corner tiles of each corner tile (these are up to four tiles). Both steps can be done with one scan, and thus in $O(N)$ time.

Now, consider the cut ℓ splitting the polyomino into subpolyominoes P_1 and P_2. Finding the corner tiles in P_1 and P_2 can be done in $O(r)$ time by a breadth-first search. We proceed with the following procedure for P_1 (analogously for P_2):

1) Get all vertical lines connecting two corner tiles in P_2 and stretch this line by one tile if a corner tile is red (this checks if a blue tile would pass a red tile).
2) Sort the set C_r of corner tiles in P_2 lexicographically by y-coordinate and then by x-coordinate.
3) Start a sweep line from bottom to top having the tiles in C_r as event points.
4) On each event point p do the following update:
   - If p is a start point of a vertical line but lies left of the current vertical line, remove p from C_r.
   - If p is a start point and lies to the right of the current line add the tile of the current line to C_r and jump to the new vertical line.
   - If p is an end point of the current vertical line, then jump to the nearest vertical line to the left and add the tile of this line to C_r.
   - If p is an end point but not of the current vertical line, remove p from C_r.
5) Repeat steps 1–4, switching left and right, to get C_l.
6) For each locally convex tile t in P_1:
   - find $q_1 \in C_r$ having highest y-coordinate below t, and $q_2 \in C_r$ having lowest y-coordinate above t. (Both shall be the left-most tile in case of ties.)
   - find $q_1' \in C_l$ having highest y-coordinate below t, and $q_2' \in C_l$ having lowest y-coordinate above t. (Both shall be the left-most tile in case of ties.)
   - If t lies to the left of segment $q_1 q_2$ and to the right of segment $q_1' q_2'$ return false.

This computes a left and right envelope of vertical lines in P_1 and P_2, respectively. This allows an easy check if there is a tile on the left/right blocking a tile from P_1 in this direction (for an example, see Fig. 9).
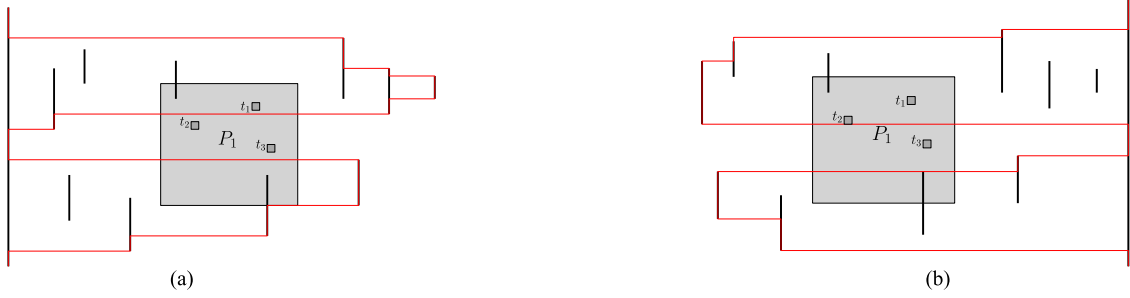
Fig. 9. Example for the data structure used in Lemma 7. We observe that t_1 is always on the wrong side of the red line and is thus blocked in both directions. Vertical lines are part of P_2.

The runtime is in $O(r \log r)$: Step 1 needs $O(r)$ time because there are $O(r)$ corner tiles and at most two vertical lines per corner tile. Sorting a set lexicographically in two dimensions can be done in $O(r \log r)$. With a careful view on step 4, we can observe that each update of the $O(r)$ event points costs $O(\log r)$ and thus in total $O(r \log r)$ time. Step 6 can be done in $O(\log r)$ time for each locally convex tile. Therefore, we need $O(r \log r)$ time in total.   □

The next theorem is straightforward to prove.

Theorem 2: Let r be the number of locally reflex tiles. We can find a valid straight 2-cut in $O(N + r^2 \log r)$ time.

Theorem 3: A decomposition tree of valid 2-cuts for a polyomino P can be used to build a labyrinth constructing P. This labyrinth can also be used for pipelining.

Proof: Consider a cycle of the seven unit steps *right, up, down, up, right, left, down*. This is the movement sequence which was already seen for convex and monotone polyominoes but with two more movements. This cycle preserves the ability to construct monotone polyominoes in the labyrinth above. Also observe that turning the gadgets seen in Figs. 4 and 5 by 90 degrees clockwise yields gadgets that put two polyominoes on top of each other.

Transforming a decomposition tree of 2-cuts for a polyomino P can easily be done: Consider the layers of the decomposition tree, with the root being layer zero, its children being layer one, and so on. In each vertex in one layer either a horizontal or vertical cut is made. Corresponding to this cut we construct a gadget putting the two children of this vertex together. At some point only monotone subpolyominoes exist. These can be build using the methods described above.

The length of a root-leaf-path may vary. In this case we can build loops so we can put two polyominoes together at the right time.   □

Theorem 4: Any straight 2-cuttable polyomino P can be build within $O(r)$ unit steps, where r is the number of locally reflex tiles in P. D copies require $O(r + D)$ unit steps.

Proof: Doing cuts along locally reflex tiles reduces the number of locally reflex tiles by at least one. This implies a maximum depth of $O(r)$ of the decomposition tree and thus, $O(r)$ cycles to produce P. As seen before, pipelining yields a construction time of $(r + D)$ unit steps, which is an amortized constant construction time if $D \in \Omega(N)$.   □

Unfortunately, the number of locally reflex tiles r can be in $\Omega(N)$ and thus, we may need $\Omega(N)$ cuts to build the polyomino. In particular, Fig. 10 left shows an example which needs $\Omega(N)$
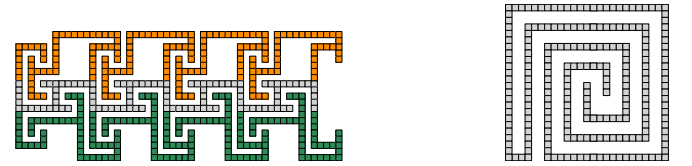

Fig. 10. Left: A polyomino needing $\Omega(N)$ steps to build as we cannot separate the green nor the orange part efficiently from the grey part. Right: Polyomino which is not 2-cuttable. Any cut splits the polyomino either in two subpolyominoes which cannot be pulled apart or into more than two subpolyominoes.
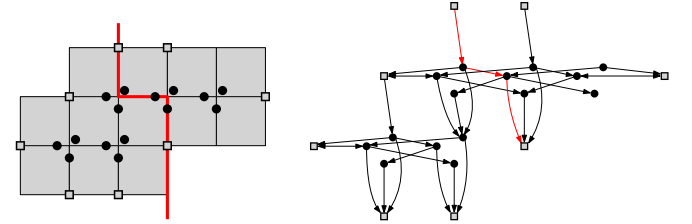

Fig. 11. A polyomino P (grey tiles) and the graph D_P (right). The vertices in V_B are shown as squares, the vertices in V_I are shown as disks. Red bold line in P is a 2-cut. Red path in the graph represents this cut.

cycles to build. Even scaling by some factor k, i.e., replacing each tile by an $k \times k$ supertile, seems not to help. Moreover, there are also polyominoes we cannot build by putting two subpolyominoes together at the same time (see Fig. 10 right).

B. Non-Straight Cuts

Considering any 2-cut makes it more difficult to find cuts, as there are exponential many possible cuts. However, we do not need to consider all cuts. For a given start s and end e on the boundary of a polyomino P, we can show that it is sufficient to consider only one cut connecting s and e. The proof is similar to the one of Lemma 5.

Theorem 5: Given a 2-cuttable polyomino P, we can find a 2-cut in time $O(r^2 N \log N)$, where r is the number of locally reflex tiles in P.

Proof: The idea of this proof is to find $O(r^2)$ 2-cuts which are then tested if they are valid. One necessary criterion is that no cut moves three units to the left or right in case of vertical cuts. This can be achieved with a directed graph D_P. As seen in Fig. 11, we add a set of $O(r)$ vertices that correspond to
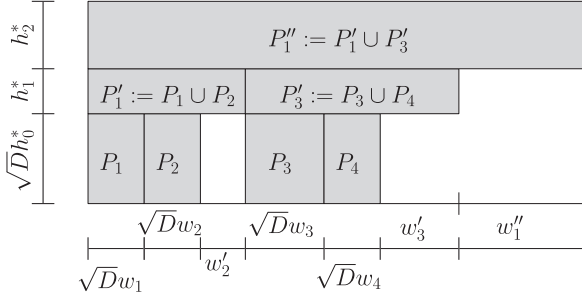
Fig. 12. Block diagram of the workspace to construct a monotone polyomino.
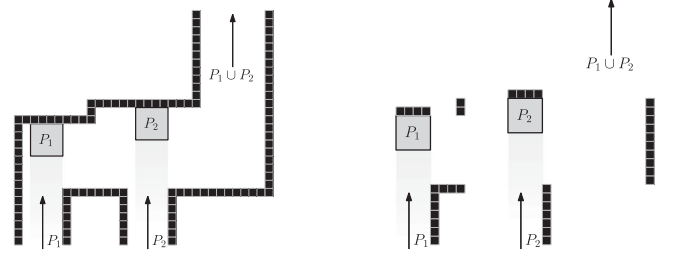

Fig. 13. Gadgets assembling two subpolyominoes. Left: With unnecessary obstacles. Right: Without unnecessary obstacles.
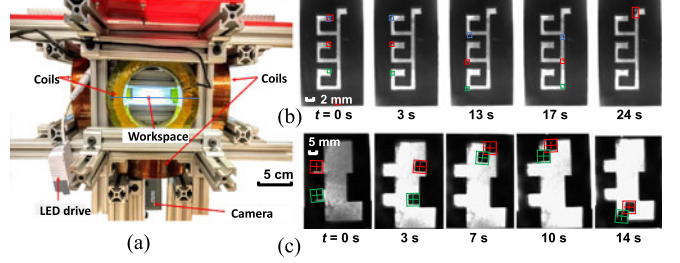

Fig. 14. (a) Magnetic manipulation workspace (b) frames from an assembly of one column of a polyomino. (c) frames from combining two polyominoes.

corners of tiles lying on the boundary of P (giving rise to the set V_B), or that correspond to corner tiles not lying on the boundary (resulting in the set V_I). We add edges between *adjacent* vertices with weight $\frac{1}{2N}$ if both vertices are in V_I. If both vertices are in V_B, then the edge has weight 2, otherwise 1.

A 2-cut is represented by a shortest path of weight at most 2.5 containing exactly two vertices of V_B. If we have at least three vertices of V_B in the shortest path, it has length at least 3. Thus, paths from one vertex in V_B to another vertex of V_B define cuts going through P. Finding all shortest paths from one vertex in V_B lasts $O(N \log N)$ time, as there are $O(N)$ edges in D_P. This implies a total time of $O(rN \log N)$ for finding all shortest paths of length at most 2.5.

Because one cut can make $O(N)$ turns, checking whether the cut is valid takes time $O(N \log N)$. Thus, checking all $O(r^2)$ cuts if they are valid needs time $O(r^2 N \log N)$. □

All techniques can be generalized for polyominoes with convex holes. However, this increases the number of possible cuts to be checked. In particular, there can be $O(r_h^3)$ possible ways to go through a hole h with r_h locally reflex tiles. Thus, the time to find a cut takes $O(N + r^3 \log r)$ using straight cuts and $O(r^3 N \log N)$ using non-straight cuts.

C. Workspace Size and Number of Obstacles

Theorem 6: Let P be a polyomino. Then, the workspace needed to assemble D copies of P can be put into a rectangle of width $O(w_P \mathcal{L}_P \cdot (\mathcal{C}_P + \sqrt{D}))$ and height $O(h_P \cdot (\mathcal{C}_P + \sqrt{D}))$, where w_P and h_P are the width and height of P, $\mathcal{C}_P$ is the number of movement steps needed, and $\mathcal{L}_P$ is the number of cuts made to decompose P into convex subpolyominoes. Furthermore, we only need $O(N(\mathcal{L}_P + \sqrt{D}))$ obstacles in the workspace.

Proof: Represent each gadget as a block. An example block diagram shown in Fig. 12 illustrates the structure of the workspace with width and height of each stage.

Consider the decomposition tree T of P induced by cuts whose leafs are convex polyominoes. For convex polyominoes we can use the construction from Lemma 6 and for each inner node of T we use the gadgets used in Theorem 1 to combine two subpolyominoes. Let $P_1, \ldots, P_k$ be the convex polyominoes in the leafs of T with width $w_1, \ldots, w_k$ and height $h_1, \ldots, h_k$. To construct one P_i, we need $O(\sqrt{D}w_i) \times O(\sqrt{D}h_0^*)$ space, where h_0^* is the maximum height of all P_i.

Now consider the j-th stage with $j \leq \mathcal{C}_P$ where some polyominoes are combined. Let P_1' and P_2' be two such polyominoes. After assembling these polyominoes the width of the polyomino P_1'' increases to $w_1'' \leq w_1' + w_2'$. Thus, the width of the workspace increases by at most $w_1' + w_2'$. We observe that any width of $P_1, \ldots, P_k$ appears at most $\mathcal{C}_P + 1$ times. With $w_i \leq w_P$ and $k \in O(\mathcal{L}_P)$, this results in a total width of $\sum_{i=1}^{\mathcal{L}_P} w_P(\sqrt{D} + \mathcal{C}_P + 1) \in O(w_P \mathcal{L}_P(\sqrt{D} + \mathcal{C}_P))$.

For the total height consider the maximum height h_j^* of all polyominoes in stage $j \leq \mathcal{C}_P$. Because we need $O(h_j^*)$ space in the vertical direction for stage j, we have have a total height of $h_P \sqrt{D} + \sum_{j=1}^{\mathcal{C}_P} h_j^* \in O(h_P(\sqrt{D} + \mathcal{C}_P))$ resulting in a rectangle of size $O(w_P(\sqrt{D} + \mathcal{C}_P)) \times O(h_P(\sqrt{D} + \mathcal{C}_P))$ enclosing the workspace.

Although the workspace may be large, the number of obstacles needed is smaller. First, ignore any obstacle not needed as a stopper (see Fig. 13). This reduces the number of obstacles to $O(w_P + h_P)$. Because $w_P, h_P \leq N$ this is $O(N)$. The same can be done for building the convex polyominoes. However, to keep the D tiles in a container we need all $O(\sqrt{D})$ obstacles. Thus, we have $O(N\sqrt{D})$ obstacles to build all convex polyominoes and $O(\mathcal{L}_P N)$ obstacles for the gadgets which is in total $O(N(\mathcal{L}_P + \sqrt{D}))$. □

V. EXPERIMENTAL DEMONSTRATION

We implemented the algorithms for staged assembly at micro and milli scale. A customized setup was used to generate a magnetic field to manipulate the magnetic particles.

1) Experimental Platform: The magnetic setup used for the experiments is shown in Fig. 14, consisting of three orthogonal pairs of coils with separation distance equivalent to the outer diameter (127.5 mm) of a coil. The coils (18 AWG, 1200 turns, Custom Coils, Inc) are actuated by six SyRen10-25 motor drivers, and a Tekpower HY3020E is used for the DC power
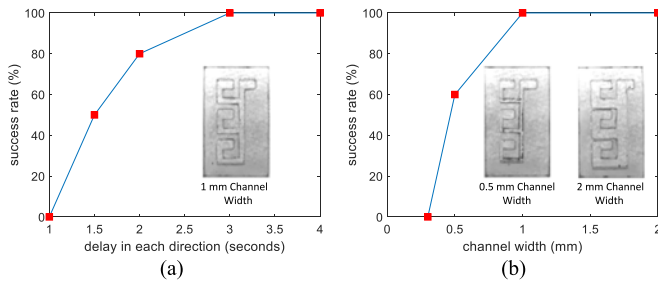
Fig. 15. Results from assembly of a micro-scale three-tile column polyomino. There are 10 trials per data point. (a) Success rate as a function of the duration control inputs were applied in each of the four directions on a workspace with 1 mm width channels. (b) Success rate as a function of channel widths using control inputs applied for 3 s in each direction.

supply. The electromagnetic platform can provide uniform magnetic fields of up to 101 G, and gradient fields up to 150 mT/m along any horizontal direction in the center of the workspace. With flux concentration cores, up to 900 mT/m gradient fields are observed in the experiment. Each flux concentration core is a solid iron cylinder 73.1 mm in diameter.

The workspaces used to demonstrate the sublinear assembly algorithms were designed to replicate the column assembly in Fig. 1 and the subpolyomino assembly in Fig. 4. Each workspace is made up of two layers of acrylic cut using a Universal Laser Cutter. The base layer is fabricated from 2 mm thick transparent acrylic, and it is glued to 5.5 mm thick acrylic, which acts as an obstacle layout. In each experiment, the workspace is placed in the center of our electromagnetic platform. The particle tiles are composed of nickel-plated neodymium cube-shaped magnets (supermagnetman.com C0010). The magnet cubes have edge lengths of 0.5 mm for micro-scale and 2.88 mm for milli-scale demonstrations. An Arduino Mega 2560 was used to control the current in the coils and the workspaces were observed with a IEEE 1394 camera, captured at 60 fps.

2) Experimental Results: In micro-scale experiments, we filled the workspaces with vegetable oil and placed a magnet cube with 0.5 mm edge length in each of the three hoppers. The workspace used in these experiments was 18 mm wide and 30 mm long. To assemble the column polyomino, a gradient magnetic field of 900 mT/m was applied in the direction sequence $\langle d, r, u, l \rangle$. Each direction input was applied for a fixed amount of time specified by a MATLAB program. A successful trial requires that all three components are joined and delivered to the top right of the workspace. Fig. 14(b) shows the completed three-tile polyomino and Fig. 15 shows representative experimental results for the assembly of the column polyomino. Successful assembly depends on the channel widths and the duration of the control inputs. Larger channel widths and longer control durations led to high success rates. Trials were always successful when the magnetic field was applied at least 3 s in each direction and when the channel width was at least 1 mm.

For milli-scale demonstrations we assembled two polyominoes, as shown in Fig. 14(c). Each polyomino is composed of four magnet cubes glued together to form a square shape. The

43 mm $\times$ 62 mm workspace was placed in a uniform, 101 G magnetic field to control the orientation of the polyominoes and then manually tilted in the direction sequence $\langle u, l, d, r, u \rangle$. See video attachment for experimental demonstrations.

VI. CONCLUSION AND FUTURE WORK

A spectrum of future work remains, most notably issues of robustness in the presence of inaccuracies, as well as the extension of our results to three-dimensional shapes. Questions in 2D include the following. Can we guarantee sublinear production times if the polyomino can be scaled by a constant? Are straight cuts sufficient, i.e., if a polyomino P is 2-cuttable, is P also straight 2-cuttable? How hard is it to decide if a polyomino cannot be built at all? Can we efficiently assemble polyomino P' that approximates P?

REFERENCES

[1] D. Arbuckle and A. A. Requicha, "Self-assembly and self-repair of arbitrary shapes by a swarm of reactive robots: Algorithms and simulations," *Auton. Robots*, vol. 28, no. 2, pp. 197–211, 2010.

[2] A. T. Becker, E. D. Demaine, S. P. Fekete, G. Habibi, and J. McLurkin, "Reconfiguring massive particle swarms with limited, global control," in *Proc. Int. Symp. Algorithms Exp. Sensor Syst., Wireless Netw. Distrib. Robot.*, 2013, pp. 51–66.

[3] A. T. Becker *et al.*, "Tilt assembly: Algorithms for micro-factories that build objects with uniform external forces," in *Proc. 28th Int. Symp. Algorithms Comput.*, 2017, vol. 92, pp. 11:1–11:13.

[4] Z. Butler, K. Kotay, D. Rus, and K. Tomita, "Generic decentralized control for lattice-based self-reconfigurable robots," *Int. J. Robot. Res.*, vol. 23, no. 9, pp. 919–937, 2004.

[5] S. Cannon *et al.*, "Two hands are better than one (up to constant factors)," in *Proc. Int. Symp. Theor. Aspects Comput. Sci.*, 2013, pp. 172–184.

[6] C. Chalk, E. Martinez, R. Schweller, L. Vega, A. Winslow, and T. Wylie, "Optimal staged self-assembly of general shapes," *Algorithmica*, vol. 80, pp. 1383–1409, 2016.

[7] H.-L. Chen and D. Doty, "Parallelism and time in hierarchical self-assembly," *SIAM J. Comput.*, vol. 46, no. 2, pp. 661–709, 2017.

[8] E. D. Demaine *et al.*, "Staged self-assembly: nanomanufacture of arbitrary shapes with O(1) glues," *Natural Comput.*, vol. 7, no. 3, pp. 347–370, 2008.

[9] E. D. Demaine, S. P. Fekete, C. Scheffer, and A. Schmidt, "New geometric algorithms for fully connected staged self-assembly," *Theor. Comput. Sci.*, vol. 671, pp. 4–18, 2017.

[10] P. S. S. Kim, A. T. Becker, Y. Ou, A. A. Julius, and M. J. Kim, "Imparting magnetic dipole heterogeneity to internalized iron oxide nanoparticles for microorganism swarm control," *J. Nanoparticle Res.*, vol. 17, no. 3, pp. 1–15, 2015.

[11] S. Manzoor, S. Sheckman, J. Lonsford, H. Kim, M. J. Kim, and A. T. Becker, "Parallel self-assembly of polyominoes under uniform control inputs," *IEEE Robot. Autom. Lett.*, vol. 2, no. 4, pp. 2040–2047, Oct. 2017.

[12] S. Vassilvitskii, J. Kubica, E. Rieffel, J. Suh, and M. Yim, "On the general reconfiguration problem for expanding cube style modular robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2002, vol. 1, pp. 801–808.

[13] J. E. Walter, J. L. Welch, and N. M. Amato, "Distributed reconfiguration of metamorphic robot chains," *Distrib. Comput.*, vol. 17, no. 2, pp. 171–189, Aug. 2004. [Online]. Available: https://doi.org/10.1007/s00446-003-0103-y

[14] E. Winfree, "Algorithmic self-assembly of DNA," Ph.D. dissertation, California Institute of Technology, Pasadena, CA, USA, 1998. [Online]. Available: http://cba.mit.edu/events/03.11.ASE/docs/Winfree.pdf

[15] Y. Zhang, X. Chen, H. Qi, and D. Balkcom, "Rearranging agents in a small space using global controls," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2017, pp. 3576–3582.