Shape Formation in a Three-dimensional Model for Hybrid Programmable Matter*

Kristian Hinnenthal[1], Dorian Rudolph[2], and Christian Scheideler[3]

1 **Paderborn University**
`krijan@mail.upb.de`
2 **Paderborn University**
`dorian@mail.upb.de`
3 **Paderborn University**
`scheideler@upb.de`

Abstract

We present the first three-dimensional model for *hybrid programmable matter*, in which small *active* robots with the computational capabilities of finite automata and very limited sensing operate on a structure of *passive* tiles. The model is an extension of the two-dimensional model of Gmyr et al. [DNA 2018], whose hexagonal tiles on the triangular lattice translate to hollow rhombic dodecahedral tiles on the face-centered cubic lattice. Following the idea of Gmyr et al., we initiate the research in this model by considering a *single* robot that can navigate through the structure, pick up tiles, and place them somewhere else to transform the structure. Contrary to the two-dimensional case, where the robot can always find a tile to move while preserving connectivity, we show that such a tile cannot be found in the three-dimensional model without an additional helper tile. We then show how the robot can construct a line with the help of such a tile in $O(n^3)$ rounds, where n is the number of tiles. A line lends itself as an intermediate structure to build more complex objects such as triangles or cubes. Our presentation is accompanied by an implementation in a 3D simulator we specifically developed for our hybrid model.

1 Introduction

Programmable matter is envisioned as a system of small particles that act in coordination to mimic a *programmable physical material* [15]. A variety of models and algorithms for programmable matter has been proposed within the last years, typically focussing on active agents with very limited computational, sensing, and movement capabilities. Notable examples are the *nubot model* [17], *metamorphic robots* [4, 16], and the *amoebot model* [7]. For the amoebot model, problems such as leader election, shape formation, coating, and shape sealing have been investigated (see, e.g., [3, 5, 6, 8]). Very recently, a hybrid variant of the amoebot model has been proposed, in which *active* agents similar to amoebots move on a structure of *passive* hexagonal tiles on the triangular lattice [7, 9, 10]. Contrary to programmable matter only comprised of active elements, here the goal is to have only few active elements that act on a potentially large structure of passive elements. As a potential application, such systems may be used to construct or repair structures in complex or hardly accessible environments such as the human body, narrow pipe systems, or space [11].

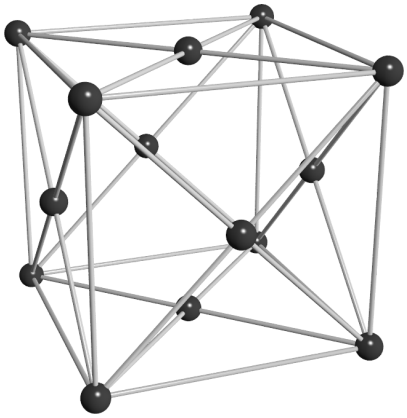
Our Contribution. We present a *three-dimensional* model for hybrid programmable matter. Whereas the three-dimensional case has been intensively studied in related areas such as modular self-reconfigurable robot systems (see, e.g., [13], or [1, 14] and the references

* Supported by DFG Project SCHE 1592/6-1 (Algorithms for Programmable Matter in a Physiological Medium).

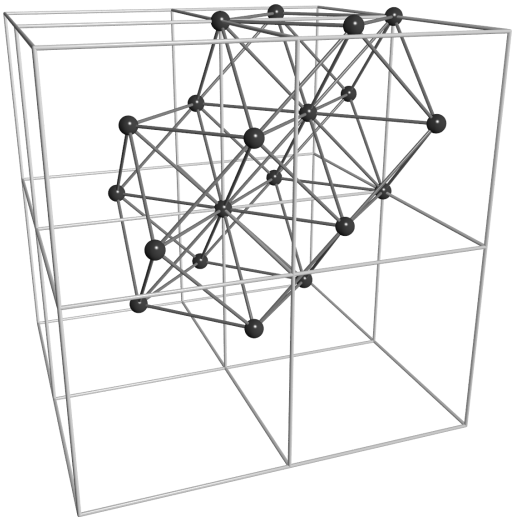
36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

therein) and molecular self-assembly (see, e.g., [12, 2]), to the best of our knowledge, there does not exist a model for our vision of programmable matter. Our model naturally extends the two-dimensional model for hybrid programmable matter presented by Gmyr et al. [10] to three dimensions: instead of robots moving on hexagonal tiles on the triangular lattice, the robots move through (hollow) rhombic dodecahedral tiles on the face-centered cubic (FCC) lattice, i.e., the adjacency graph of the FCC sphere-packing (see Fig. 1).

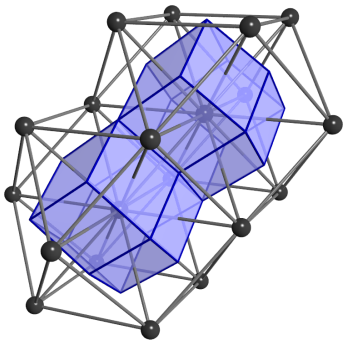
Note that the space-filling tesselation of the rhombic dodecahedron is precisely the Voronoi tesselation of the FCC lattice. Compared to cubic structures as an apparent alternative space-filling tiling, the rhombic dodecahedral structures are stronger connected and therefore more sturdy, as every node has 12 neighbors and each pair of adjacent cells shares a common face. Furthermore, this allows robots to move from a tile to each adjacent tile through the incident face and without having to leave the tile structure. Note that whereas in the two-dimensional model tiles can be moved by carrying them *over* the tile structure, we require the robot to carry tiles *through* existing tiles. Therefore, in a practical setting, the robot needs to be able to fold and unfold carried tiles to squeeze them through other tiles.



(a) Subgraph of the FCC lattice that corresponds to a single cube.



(b) Larger subgraph of the FCC lattice (contained within 8 cubes).



(c) Two voronoi cells (blue rhombic dodecahedra) of the subgraph of (b).

Figure 1 Rhombic dodecahedral tiles in the FCC lattice.

We follow the idea of Gmyr et al. and initiate the research in this model by considering the problem of forming a line with a *single* robot. Specifically, we present an algorithm that transforms any initially connected structure into a line in $O(n^3)$ rounds while preserving connectivity at all times. Similar to the two-dimensional case, a line can be used as an intermediate structure to build a triangle, and simple three-dimensional structures such as pyramids and cubes can be constructed in a very similar fashion; due to space constraints, we leave out the exact description of these algorithms. As our results indicate, moving tiles in three dimensions without violating connectivity requires much more sophisticated behavior. Specifically, we show that the robot cannot locally decide which tile can be moved *at all* without violating connectivity, which we address by providing the robot with an additional helper tile. Our algorithm has been implemented in a three-dimensional hybrid programmable matter simulator that we developed for visualization and evaluation purposes. A few screenshots of the formation of a line, as well as the transformation of the line into a pyramid, can be found in Fig. 2.

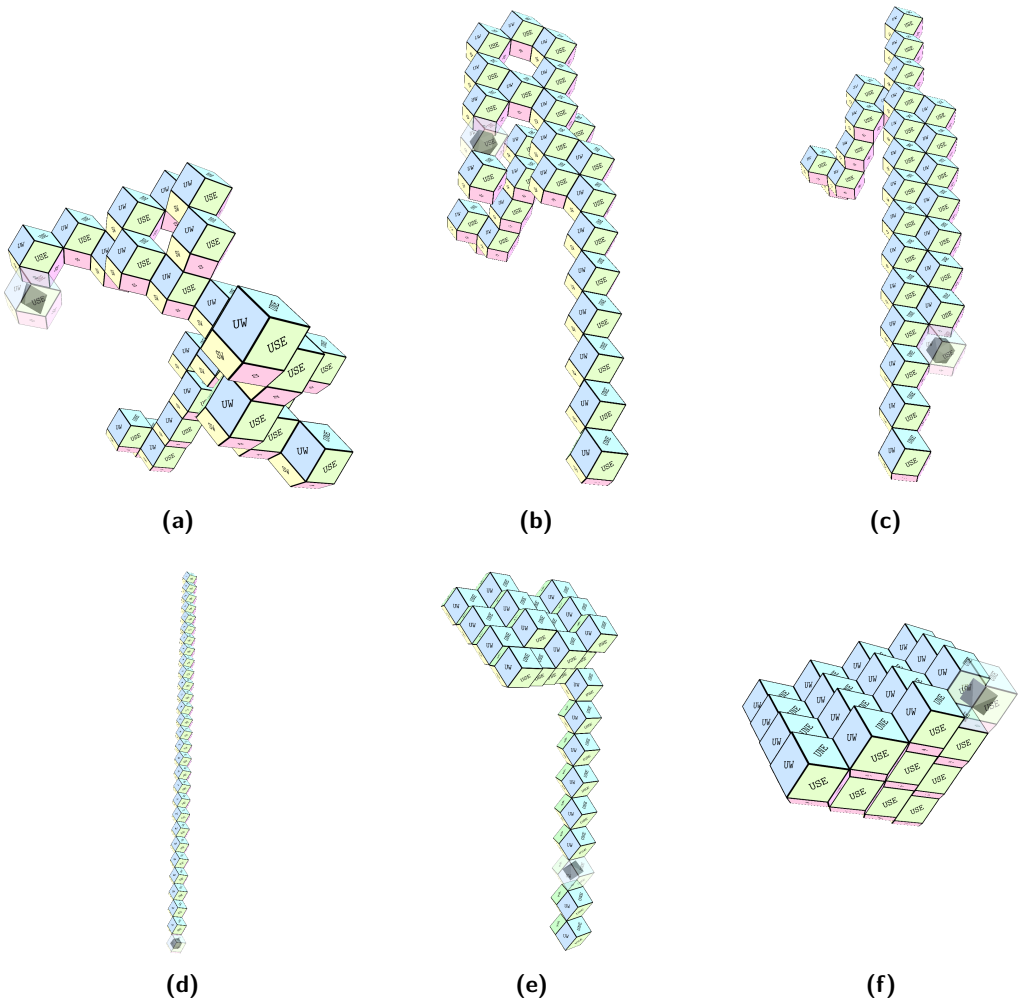


Figure 2 Simulator screenshots of constructing a line (d) and transforming it into a pyramid (f).

Model. As in the two-dimensional model [10], we assume that a single robot r operates on a finite set of n tiles. In our model, each tile forms a hollow rhombic dodecahedron and occupies exactly one node of the infinite FCC lattice $G = (V, E)$. For ease of presentation, we regard G as a stack of hexagonal layers, where each layer can be represented in the triangular lattice (see gray scale layers in Fig. 3).

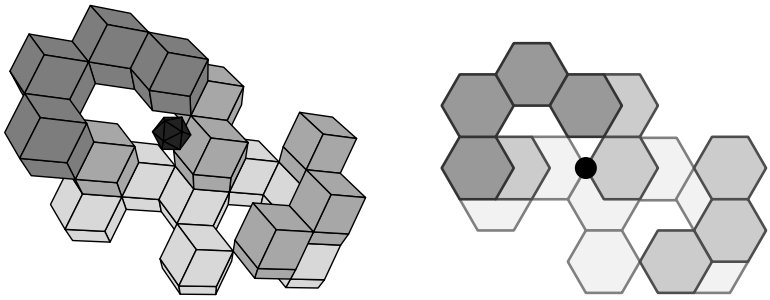


Figure 3 Example configuration with three layers depicted in 3D (left) and as a stack of hexagonal structures in the corresponding triangular lattices (right). The robot is shown as a black circle.

A *configuration* (T, p) consists of a set $T \subseteq V$ of all nodes occupied by tiles, and the robot's position $p \in V$. We assume that the initial position of the robot is occupied by a tile and that the robot initially *carries* one of the n tiles. Every node $u \in V$ is adjacent to twelve neighbors, and, as indicated in Fig. 4, we describe the relative positions of adjacent nodes by the cardinal directions N, NE, SE, S, SW, NW, as well as the "up-directions" USE, UNE, UW, and the "down-directions" DE, DNW, and DSW. The robot must always be on or adjacent to a node occupied by a tile. Additionally, if the robot does not carry a tile, we require the subgraph of G induced by T to be connected; otherwise, the subgraph induced by $T \cup \{p\}$ must be connected. This restriction prevents the configuration from falling apart in a practical scenario.

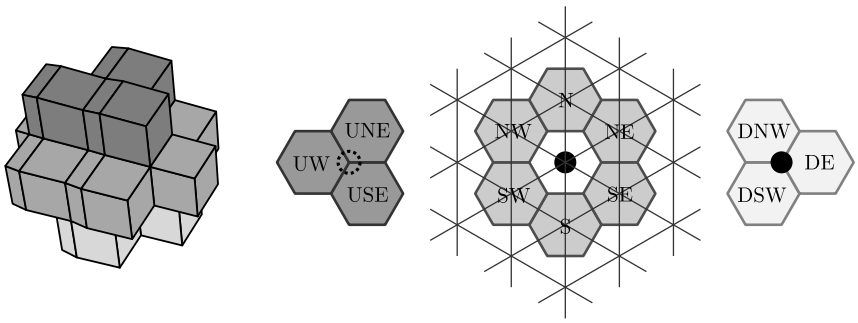


Figure 4 Tiles are labeled with the direction from the robot's point of view. The three layers are depicted separately. The center one also shows the grid.

The robot has only constant memory, which gives it the computational capabilities of a finite automaton, and operates in rounds of *Look-Compute-Move* cycles. In the *Look* phase of a round, the robot can observe its node p and the twelve neighbors of that node. For each of these nodes, it can determine whether the node is occupied or not. In the *Compute* phase, the robot may change its internal state (or terminate) and determines its next move according to the observed information. In the *Move* phase, the robot can either (1) pick up a tile from p, if $p \in T$, (2) place a tile it is carrying at p if $p \notin T$, or (3) move to an adjacent

node while possibly carrying a tile with it. The robot can carry at most one tile.

2 Safely Movable Tiles

As shown by Gmyr et al. for the two-dimensional model [10, Theorem 1], a single robot cannot find a *safely removable tile*, i.e., a tile on a node v such that the subgraph induced by $T \setminus \{v\}$ is connected. Gmyr et al. circumvent this issue by repeatedly locating and moving a *safely movable tile* to transform the structure, i.e., a tile at some node v for which there exists an adjacent node v' such that the subgraph induced by $(T \setminus \{v\}) \cup \{v'\}$ is connected. Specifically, the robot moves tiles to adjacent nodes without even violating connectivity in the tile's *local* neighborhood, which obviously also preserves global connectivity. Formally, we call a tile at node v *locally safely movable* if it has an adjacent node v' such that the graph induced by $N(v) \cup \{v'\}$ is connected, where $N(v) \subseteq T$ is the set of occupied nodes adjacent to v.

As an example, in two dimensions, a hollow hexagon of tiles can be transformed into a triangle by repeatedly moving locally safely movable tiles (i.e., the hexagon's corners) inwards until there is no hole anymore [10]. At this point, it is very easy to find safely removable tiles and rearrange them into the target structure (e.g., a triangle). Since a safely removable tile cannot be found in general [10, Theorem 1], the strategy of moving locally safely movable tiles to create an intermediate structure proves to be very useful. However, the impossibility result for safely removable tiles does not only translate to the three-dimensional case as well, there even exists a configuration that does not have *any* locally safely movable tile in our model (see Fig. 5), making it impossible to use the approach for two dimensions. Even worse, the following theorem shows that the robot cannot find a safely movable tile in general, even though it might exist.

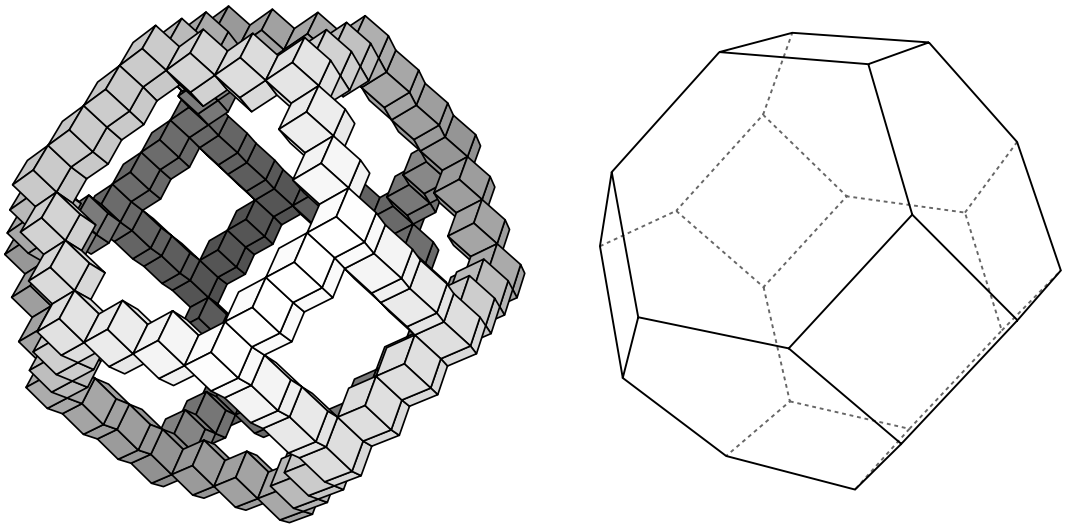


Figure 5 A tile configuration without any *locally* movable tile (i.e., a tile that can be moved without disconnecting its local neighborhood) and the truncated octahedron that it resembles. Note that whereas each tile is in fact safely movable, this cannot be decided locally by the robot.

▶ **Theorem 2.1.** *There does not exist a robot that terminates on a safely movable tile on every initial configuration without ever moving a tile.*

Proof sketch. Construct a tree G (i.e., the graph induced by T is a tree) whose only safely movable tiles are its leaves, and which a robot cannot distinguish from a variation H of Fig. 5 with sufficiently long edges. Roughly speaking, a finite automaton robot cannot "measure" the length of a line, therefore the robot will behave exactly the same on G as it does on H. Since it terminates on H after having traversed a constant number of edges (otherwise it would enter an infinite loop), it must also terminate on G before having reached its leaves (which are the only safely movable tiles in G). ◀

This implies that it is impossible for the robot to select a tile to move in general, which justifies our choice to equip the robot with an initial tile. As we show in the following section, the robot can use the additional tile to transform the structure without violating connectivity.

3 Line Formation Algorithm

In this section, we present our algorithm to transform any initial configuration into a line in $O(n^2 \cdot h) \subseteq O(n^3)$ rounds, where h is the height (i.e., number of layers) of the initial tile configuration. The basic idea is to repeatedly merge *columns*, i.e., maximal lines of occupied nodes in the N/S direction, until only a single column remains. The robot cycles through the following phases, starting with phase *FC*:

- ***FC*** (find column): Find a column without neighbors (i.e., adjacent tiles) above (i.e., USE, UNE or UW) or west (i.e., NW or SW) by traversing columns from N to S and moving up or west whenever possible. Eventually, the robot will find such a column. If that column has a neighbor below (i.e., DE, DNW or DSW), switch to *MCD*; if otherwise it has a neighbor to the east (i.e., NE or SE), switch to *MCE*. Halt once only a single column remains.
- ***MCD*** (move column down): The goal of this phase is to move the column's northernmost remaining tile to the first (from N to S) empty node DE of the column, or, if no such node exists, S of the column. Switch back to *FC* once either all tiles have been moved DE, or the column was merged with another to the south.
- ***MCE*** (move column east): Same as *MCD*, but move tiles SE instead of DE.

We next describe the technically more challenging phases *MCD* and *MCE* in more detail. Our algorithm requires the robot to basically always carry a tile with it, therefore moving a tile t to some node u is actually performed by first placing the carried tile at u and picking up t afterwards (where the movement from u to t is the only time the robot does *not* carry a tile). In the following, let C denote the set of nodes occupied by the column the robot r currently operates on. C_N and C_S denote the column's northernmost and southernmost nodes, respectively. For any node u, denote by, e.g., $u + \mathrm{NW}$ the node NW of u.

***MCD*:** At the beginning of each iteration of this phase the robot's position is $p = C_\mathrm{N} + \mathrm{NE}$ (see Fig. 6a). Recall that before moving the tile t' at C_N, r first has to place its carried tile t at the first empty node DE of C, and then picks up t'. To do so, it follows the path depicted in Fig. 6a until it finds an empty node, and places t. Afterwards, r moves back to C_N and picks up t' (see Fig. 6b–6c). r then distinguishes the following cases:

- If t was placed at $v = C_\mathrm{S} + \mathrm{S}$, and there was a tile at $v + \mathrm{S}$ (see Fig. 6h), r moves to an adjacent tile and switches to *FC* (see Fig. 6i). In this case, r has essentially merged C with the column to its south.
- Else, if r's position is $p = C_N = C_S$, which is the case if $p + \mathrm{S} \notin T$, then r has picked up the column's last tile. The robot then moves to an adjacent tile and switches to *FC*.
- Otherwise, r moves SE and begins the next iteration of phase *MCD* (see Fig. 6c).

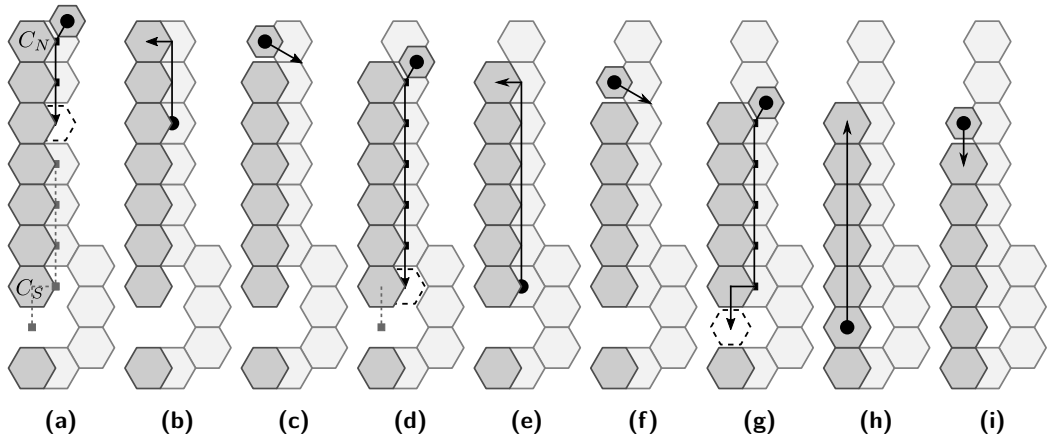


Figure 6 Example of the phase *MCD*. In (a), (d), and (g), r follows the black path until it reaches the first empty node (marked with a dashed outline). The dashed line indicates how the path would proceed if r had not already found an empty node. The carried tile is shown smaller.

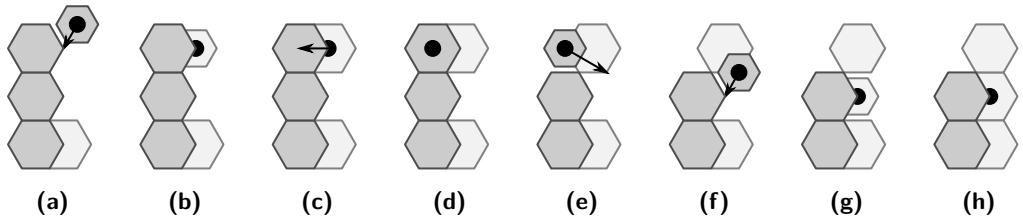


Figure 7 Maintaining connectivity during phase *MCD* using the carried tile.

Fig. 7 illustrates how r makes use of the initially carried tile to maintain connectivity when moving tiles DE (see steps 7e–7g). Note that a new iteration of the phase starts at 7f, where r being at $C_{\mathrm{N}} + \mathrm{NE}$ is essential for the configuration's connectivity.

***MCE*:** The behavior of r in this phase only differs from *MCD* in the path r follows to find the first empty node SE of C (see Fig. 8a). The depicted example shows the case where r moves all tiles SE without merging C with another column to the south (as in Fig. 6).

Using the observation displayed in Fig. 7, it is straightforward to show the next lemma.

▶ **Lemma 3.1.** *The robot does not disconnect the configuration.*

It remains to be proven that r does indeed build a line in $O(n^2 \cdot h)$ rounds.

▶ **Lemma 3.2.** *The robot spends at most $O(n^2)$ rounds in the phases* MCE *and* MCD*.*

Proof sketch. The robot r only moves tiles in the directions S, SE, and DE (assuming the previously discussed interpretation of moving tiles). One can show that if r moves tiles by a total distance of k in phase *MCE* or *MCD*, it spends $O(k)$ rounds in that phase. It can further be shown that the distance between each tile's initial and final positions is bounded by $O(n)$. Hence, tiles are moved a total distance of $O(n^2)$. ◀

▶ **Lemma 3.3.** *The robot spends at most $O(n^2 \cdot h)$ rounds in phase* FC*.*

Proof sketch. r traverses each column at most once during *FC*, since it only exits columns by moving west or up. Hence, it exits *FC* after $O(n)$ rounds. *MCE* always merges C with another column (either S or SE of C), thereby reducing the total number of columns. *MCD*

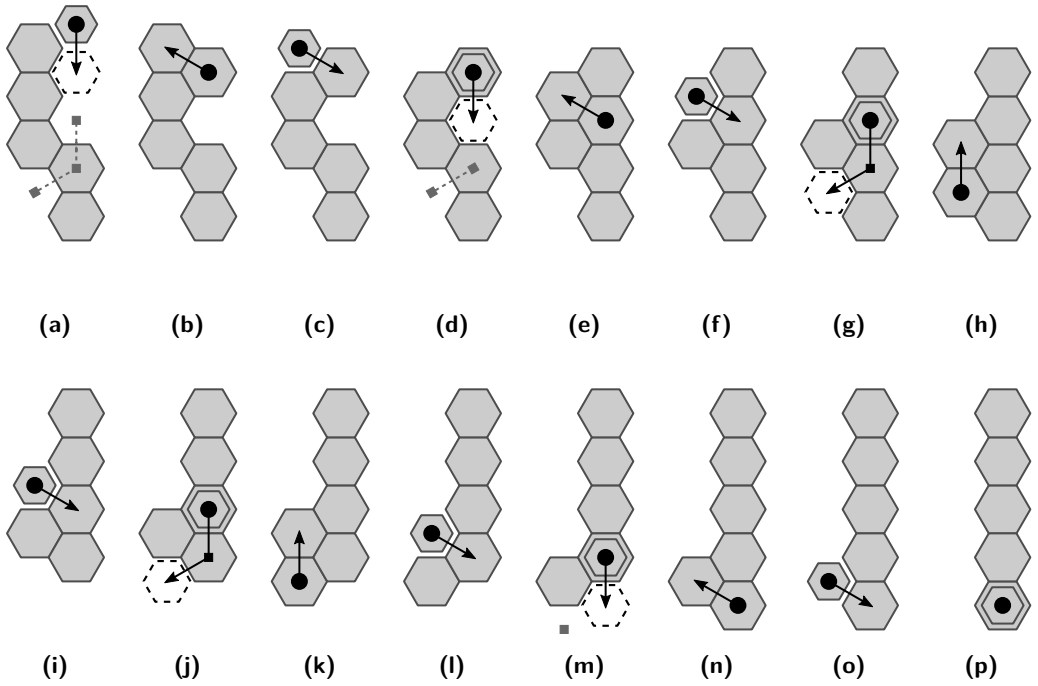


Figure 8 Example of the phase *MCE*.

merges C with another column to its S, or it moves at least one tile down. The former case occurs at most $O(n)$ times since there are at most $O(n)$ columns. The latter occurs at most $O(n \cdot h)$ times, since r only moves a tile down if its column already has a neighbor below. Hence, r enters phases *MCE* and *MCD* (and thus also *FC*) at most $O(n \cdot h)$ times. ◀

Since r only halts once it has built a line, we conclude the following theorem.

▶ **Theorem 3.4.** *The robot builds a line in $O(n^2 \cdot h)$ rounds.*

Acknowledgments. The authors would like to thank Irina Kostitsyna for many helpful discussions from which this work originated.

References

1 Hossein Ahmadzadeh, Ellips Masehian, and Masoud Asadpour. Modular Robotic Systems: Characteristics and Applications. *Journal of Intelligent and Robotic Systems: Theory and Applications*, 81(3-4):317–357, 2016.

2 Faisal A. Aldaye, Alison L. Palmer, and Hanadi F. Sleiman. Assembling materials with dna as the guide. *Science*, 321(5897):1795–1799, 2008.

3 Rida A. Bazzi and Joseph L. Briones. Stationary and Deterministic Leader Election in Self-organizing Particle Systems. In *International Symposium on Stabilizing, Safety, and Security of Distributed Systems*, pages 22–37, 2019.

4 Gregory S. Chirikjian. Kinematics of a metamorphic robotic system. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation (ICRA)*, volume 1, pages 449–455, 1994.

5 Joshua J. Daymude, Zahra Derakhshandeh, Robert Gmyr, Alexandra Porter, Andréa W. Richa, Christian Scheideler, and Thim Strothmann. On the runtime of universal coating for programmable matter. *Natural Computing*, 17:81–96, 2018.

6 Joshua J. Daymude, Robert Gmyr, Kristian Hinnenthal, Irina Kostitsyna, Christian Scheideler, and Andréa W. Richa. Convex Hull Formation for Programmable Matter. In *Proc. of the 21st International Conference on Distributed Computing and Networking (ICDCN) (to appear)*, 2020.

7 Joshua J. Daymude, Kristian Hinnenthal, Andréa W. Richa, and Christian Scheideler. Computing by programmable particles. In *Distributed Computing by Mobile Entities: Current Research in Moving and Computing*, pages 615–681. 2019.

8 Zahra Derakhshandeh, Robert Gmyr, Thim Strothmann, Rida Bazzi, Andréa W. Richa, and Christian Scheideler. *Leader Election and Shape Formation with Self-organizing Programmable Matter*, pages 117–132. 2015.

9 Robert Gmyr, Kristian Hinnenthal, Irina Kostitsyna, Fabian Kuhn, Dorian Rudolph, and Christian Scheideler. Shape recognition by a finite automaton robot. In *43rd International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 52:1–52:15, 2018.

10 Robert Gmyr, Kristian Hinnenthal, Irina Kostitsyna, Fabian Kuhn, Dorian Rudolph, Christian Scheideler, and Thim Strothmann. Forming tile shapes with simple robots. *Natural Computing*, 2019.

11 Benjamin Jenett and Daniel Cellucci. A mobile robot for locomotion through a 3d periodic lattice environment. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5474–5479, 2017.

12 Matthew R. Jones, Robert J. Macfarlane, Byeongdu Lee, Jian Zhang, Kaylie L. Young, Andrew J. Senesi, and Chad A. Mirkin. DNA-nanoparticle superlattices formed from anisotropic building blocks. *Nature Materials*, 9(11):913–917, 2010.

13 Ara N. Knaian, Kenneth C. Cheung, Maxim B. Lobovsky, Asa J. Oines, Peter Schmidt-Neilsen, and Neil A. Gershenfeld. The Milli-Motein: A self-folding chain of programmable matter with a one centimeter module pitch. In *IEEE International Conference on Intelligent Robots and Systems*, pages 1447–1453, 2012.

14 Esben Hallundbæk Østergaard, Kristian Kassow, Richard Beck, and Henrik Hautop Lund. Design of the atron lattice-based self-reconfigurable robot. *Autonomous Robots*, 21(2):165–183, 2006.

15 Tommaso Toffoli and Norman Margolus. Programmable matter: Concepts and realization. *Physica D: Nonlinear Phenomena*, 47(1):263–272, 1991.

16 Jennifer E. Walter, Elizabeth M. Tsai, and Nancy M. Amato. Algorithms for fast concurrent reconfiguration of hexagonal metamorphic robots. *IEEE Transactions on Robotics*, 21(4):621–631, 2005.

17 Damien Woods, Ho-Lin Chen, Scott Goodfriend, Nadine Dabby, Erik Winfree, and Peng Yin. Active self-assembly of algorithmic shapes and patterns in polylogarithmic time. In *Proceedings of the 4th Conference on Innovations in Theoretical Computer Science*, ITCS '13, pages 353–354, 2013.