Recognition and Reconfiguration of Lattice-Based Cellular Structures by Simple Robots

Amira Abdel-Rahman[1], Aaron T. Becker[2], Daniel E. Biediger[2], Kenneth C. Cheung[3], Sándor P. Fekete[4], Benjamin Jenett[1,3], Eike Niehs[4], Christian Scheffer[4], Arne Schmidt[4], and Mike Yannuzzi[2]

1 **Center for Bits and Atoms (CBA), Massachusetts Institute of Technology, Cambridge, MA, USA. bej@mit.edu, amira.abdel-rahman@cba.mit.edu**
2 **Department of Electrical and Computer Engineering, University of Houston, USA. {atbecker,dbiediger}@uh.edu**
3 **NASA Ames Research Center, Coded Structures Lab (CSL), Moffett Field, CA, USA. kenny@nasa.gov**
4 **Department of Computer Science, TU Braunschweig, Germany. {s.fekete, e.niehs, c.scheffer, arne.schmidt}@tu-bs.de**

Abstract

We consider recognition and reconfiguration of lattice-based cellular structures by very simple robots with only basic functionality. The underlying motivation is the construction and modification of space facilities of enormous dimensions, where the combination of new materials with extremely simple robots promises structures of previously unthinkable size and flexibility. We present algorithmic methods that are able to detect and reconfigure arbitrary polyominoes, based on finite-state robots, while also preserving connectivity of a structure during reconfiguration. Specific results include methods for determining a bounding box, scaling a given arrangement, and adapting more general algorithms for transforming polyominoes.

1 Introduction

Building and modifying large-scale structures is an important and natural objective in a vast array of applications. In many cases, the use of autonomous robots promises significant advantages, but also a number of additional difficulties, in particular when aiming for construction in orbit around earth. In recent years, a number of significant advances have been made to facilitate overall breakthroughs. One important step has been the development of ultra-light and scalable composite lattice materials [16] that allow the construction of modular, reconfigurable, lattice-based structures [18]. A second step has been the design of simple autonomous robots [17,19] that are able to move on the resulting lattice structures and move their elementary cell components, thereby allowing the reconfiguration of the overall edifice.

In this paper, we address the next step in this hierarchy: Can we enable extremely simple robots to perform a more complex construction task for cellular structures in space, such as patrolling and marking the perimeter, scaling up a given seed construction, and a number of other design operations? As we demonstrate, even the extremely limited capabilities of machines with a finite number of states suffice for these tasks. In particular, we show that two robots suffice to construct the bounding box for a given arrangement, without losing connectivity. This is then used for other objectives, such as scaling up a given arrangement by a constant factor, as well as other, more general transformations.

1.1 Related Work

Assembly by simple robots has also been considered at the micro scale, where global control is used for supplying the necessary force for moving agents, e.g., see Becker et al. [3] for

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.

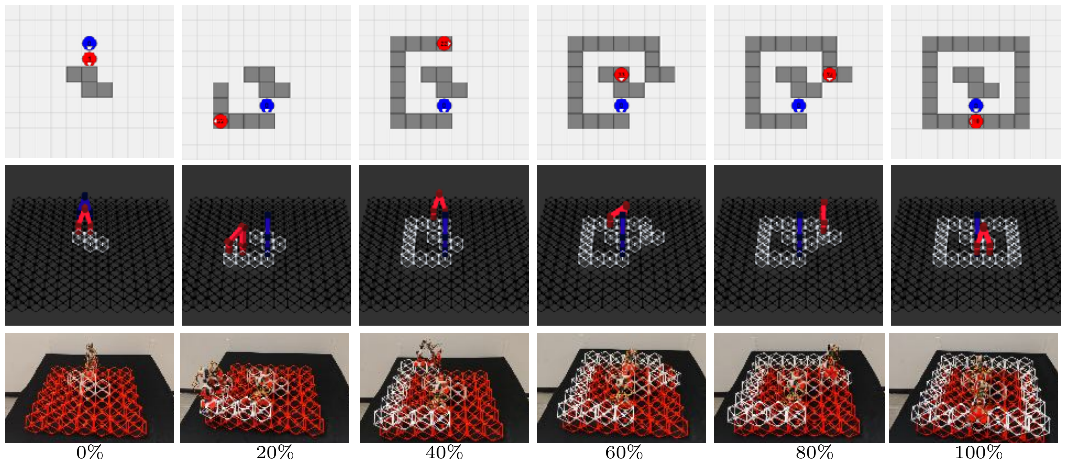

Figure 1 Snapshots from building a bounding box for a z-shaped polyomino using a 2D simulator, a 3D simulator, and staged hardware robots; shown are steps {0, 24, 48, 72, 96, 120}. See our video [1] for more context and animations.

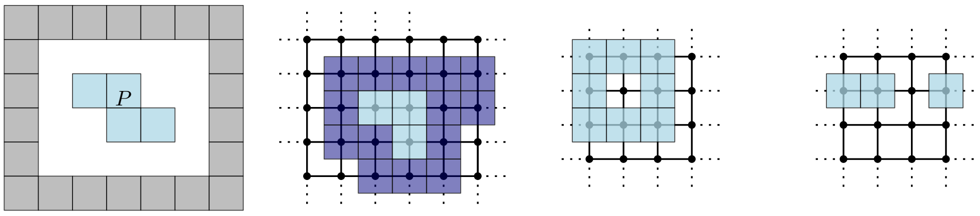

Figure 2 From left to right: Bounding box (gray) surrounding a polyomino (blue); the boundary ∂P (dark blue) of a shape P; a non-simple polyomino with one hole; a disconnected arrangement.

the corresponding problem of motion planning, Schmidt et al. [20] for using this model for assembling structures, and Balanza-Martinez et al. [2] for theoretical characterizations.

From an algorithmic view, we are interested in *different models representing programmable matter* and further recent results. Inspired by the single-celled amoeba, Derakhshandeh et al. introduced the Amoebot model [6, 10]. The Amoebot model provides a framework based on an equilateral triangular graph and active particles that can occupy a single vertex or a pair of adjacent vertices within that graph. Further related work to the amoebot model can be found in [4, 5, 7–9, 11]. In [15], Gmyr et al. introduced a model with two types of particles: active robots acting like a deterministic finite automaton and passive tile particles. Further results are shown in [14]. Fekete et al. [12] introduced more complex geometric algorithms for copying, reflecting, rotating, and scaling a given polyomino as well as an algorithm for constructing a bounding box surrounding a polyomino. However, their algorithms do not guarantee connectivity of intermediate arrangements. We build upon their model by allowing multiple robots working on the same grid environment.

2 Preliminaries

We consider an infinite *square grid graph* G, where $\mathbb{Z}^2$ defines the *vertices*, and for every two vertices with distance one there is a corresponding *edge* in G. We use the compass directions (N, E, S, W) for orientation when moving on the grid and may use *up, right, down*, and *left* synonymously.

Every vertex of G is either *occupied* by a tile or *unoccupied*. *Tiles* represent passive particles of programmable matter that cannot move or manipulate themselves. The maximal connected set of occupied vertices is called *polyomino*.

The *boundary* of a polyomino P is denoted by ∂P and includes all tiles of P that are (horizontally, vertically or diagonally) adjacent to an empty vertex (see also Figure 2). Polyominoes can have *holes*, i.e., finite maximal connected sets of empty vertices. Polyominoes without holes are called *simple*; otherwise, they are *non-simple*. The bounding box of a given polyomino P is defined as the boundary of the smallest rectangle enclosing P enlarged by one unit; it will be denoted by $bb(P)$ (see Figure 2).

We use *robots* as active particles in our model. These robots work like *finite deterministic automata* that can move around on the grid and manipulate the polyomino. A robot has the abilities to move along the edges of the grid graph and to change the state of the current vertex by placing or removing a tile on it. The robots work in a series of Look-Compute-Move (LCM) steps. Depending on the current state of the robot and the vertex it is positioned on (Look), the next step is computed according to a specific transition function δ (Compute), which determines the future state of robot and vertex, and the actual movement (Move). In the move phase, the robot can either remove a tile, place a tile (if there is not already a tile) or move to a adjacent vertex. *Moving* a tile is the same as deleting the tile at the start and placing a tile after the move sequence. In the case of multiple robots, we assume that they cannot be placed on the same vertex at the same time. Communication between robots is limited to adjacent vertices and can be implemented by expanding the Look phase by the states of all adjacent robots.

Connectivity is ensured if the union of all placed tiles and all used robots is connected. Accordingly, a robot can hold two components together (see blue robot in Figure 3).

3 Constructing a Bounding Box

In this section, we describe an algorithm to construct the bounding box while keeping connectivity of intermediate arrangements. Due to space constraints, we only sketch technical details; see the full version of the paper [13] for a full description. To accomplish the required connectivity, we specify, without any loss of generality, that the connection between $bb(P)$ and P must be on the south side of the boundary. For ease of presentation, the polyomino is shown in blue and the bounding box in gray; the robots cannot actually distinguish between those tiles. In the following, we assume that two robots are placed adjacent to each other on an arbitrary tile of the polyomino P, and that the first robot R_1 (shown in red in all figures) is the leader. As we will see, the second robot R_2 (blue in all figures) holds the polyomino and the bounding box together. In practice, we would rather use a special marker, called pebble, to mark a tile that holds P and $bb(P)$ together.

The construction can be split into three phases: (1) finding a start position, (2) constructing the bounding box, and (3) the clean-up. To find a suitable start position, we search for a locally y-minimal vertex that is occupied by a tile. This can be done by scanning the current row (i.e., moving left until we find an empty vertex and then moving right) and moving downwards whenever possible. Afterwards, R_1 starts the bounding box construction one vertex further down. This brings us to phase (2).

The construction of the bounding box is performed clockwise around P, i.e., whenever possible, R_1 makes a right turn. At some point, R_1 finds a tile either belonging to P or to the bounding box. To decide whether a tile t belongs to P or the current bounding box, we start moving around the boundary of the shape t belongs to. At some point, R_1 reaches R_2. If R_1 is above R_2 then t is a tile of P, otherwise t is a tile of the bounding box. To find

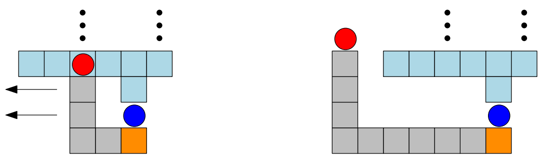

Figure 3 Left: R_1 (red) hits a tile belonging to P. Right: The triggered shifting process is finished.

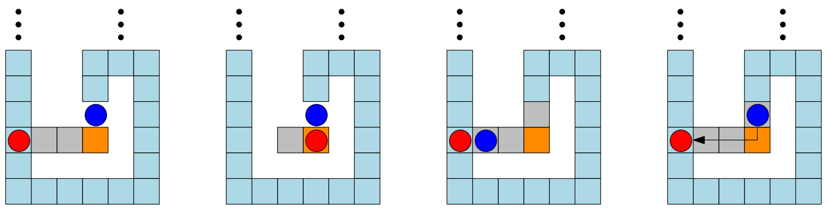

Figure 4 Traversing a gap by building a bridge. From left to right: R_1 finds a particle t not belonging to $bb(P)$. R_1 then picks up R_2, so both can move to t. Both, R_1 and R_2 reached t. Afterwards, R_2 deletes remaining pieces of the old bounding box.

t again, we move below R_2 and follow the construction until we cannot move any further. From there we can carry on building $bb(P)$.

Now, consider the two cases: If the tile does not belong to P, we are done with phase (2) and can proceed to phase (3). If it is a tile belonging to P, we need to shift the current line outwards until there is no more conflict, then continue the construction (see Figure 3). If the line to shift is the first line of the constructed bounding box, we know that there exists a tile of P that has the same y-coordinate than the current starting position. Therefore, we build a bridge to traverse this gap, as shown in Figure 4. Afterwards, we can restart from phase (1).

For phase (3), consider the case when R_1 reaches a tile from the bounding box. If the hit tile is not a corner tile, the current line needs to be shifted outwards until the next corner is reached (see Figure 5(a)). Then we can search for another suitable connection between P and $bb(P)$, place a tile there, and get to R_2 to remove unnecessary parts of the bounding box (see Figure 5(b)-(d)). To move to R_2, we move counterclockwise around $bb(P)$ until we find a tile with three adjacent tiles. From there we move north and follow the path until we find R_2. Because $bb(P)$ has only one tile with three adjacent tiles left, we can always find the connection between P and $bb(P)$.

▶ **Theorem 1.** *Given a polyominino P of width w and height h, building a bounding box surrounding P with the need that boundary and P are always connected, can be done with two robots in $O(\max(w,h)\cdot(wh+k\cdot|\partial P|))$ steps, where k is the number of convex corners in P.*

The proof of this theorem is analogous to that from [12]; see [13] for full details.

4 Scaling Polyominoes

Now we consider scaling a given shape by a factor c by building a scaled copy to the left of the bounding box. This copy process will be done column-wise from right to left. In the following we assume that the robot R_1 already built the bounding box and is positioned on one of its tiles.

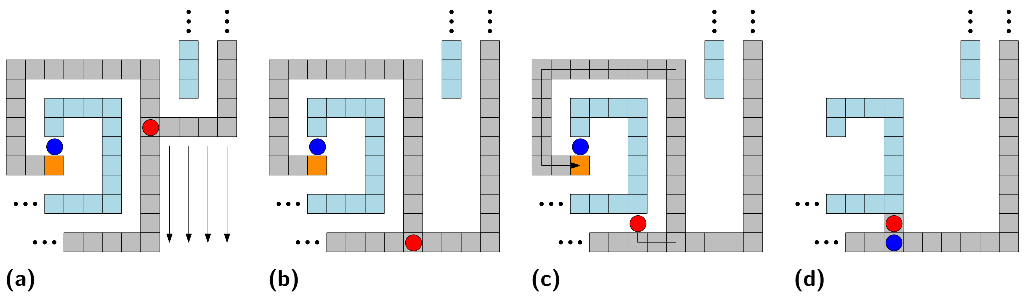

Figure 5 The second case of finishing the bounding box. (a) An already constructed part of the bounding box is hit. (b) The last boundary side is shifted. (c) R_1 found a suitable new connectivity vertex above the southern side, places a tile and retraces its path to the initial starting position. (d) The unnecessary part of the bounding box is removed and both robots catch up to the new connection.

4.1 Scaling

The scaling process can be divided into two phases: (1) the preparation phase, and (2) the scaling phase. In phase (1) we fill up the last column within $bb(P)$, add a tile in the second last column above the south side of $bb(P)$ and remove the lowest tile (called *column marker*) and third lowest tile (called *row marker*) on the east side of $bb(P)$ (see Figure 6). This gives us three columns within the bounding box (including $bb(P)$ itself). The first (from west to east) is the current column of P to scale. The second column, which is filled with tiles excepting the topmost row, is used to ensure connectivity and helps to recognize the end of the current column. The third column marks the current overall progress, i.e., we can find the tile in the correct current column and row that we want to scale next; we only have to move two steps to the west from the row marker to find the next vertex to scale.

In phase (2), we simply search for the vertex v to scale by moving to the column marker, then to the row marker, and afterwards moving two steps to the west. We then place the row marker one vertex upwards. For possible cases, see Figure 6. When we reach the top row of the bounding box, we move the column marker one vertex to the left and place a new row marker. Then we add a $c \times c$ square to the left of $bb(P)$, if v was occupied. If we did not move the column marker, we move left from the south side of $bb(P)$ until we reach an end and start moving up until we find the place to build the $c \times c$-square. If v was empty, then we leave out one tile within the square, e.g. the tile in the middle (see Figure 6 right).

After scaling a column that only contained empty vertices, we know that we are done with scaling. Thus, we can start removing all tiles, proceeding columnwise within $bb(P)$ from right to left. If necessary, all scaled empty tiles can also be removed by one scan through the scaled field. A formal proof can be found in the full version [13].

▶ **Theorem 2.** *After building $bb(P)$, scaling a polyomino P of width w and height h by a constant scaling factor c without loss of connectivity can be done with one robot in $O(wh \cdot (c^2 + cw + ch))$ steps.*

4.2 Adapting Algorithms

As shown in [12], there are algorithms that may not guarantee connectivity. An immediate consequence of being able to scale a given shape is that we can simulate any algorithm $\mathcal{A}$ within the Robot-on-Tiles model while guaranteeing connectivity: We first scale the polyomino by three and then execute $\mathcal{A}$ by always performing three steps into one direction if $\mathcal{A}$ does one step. If at some point the robot needs to move through empty vertices, then we

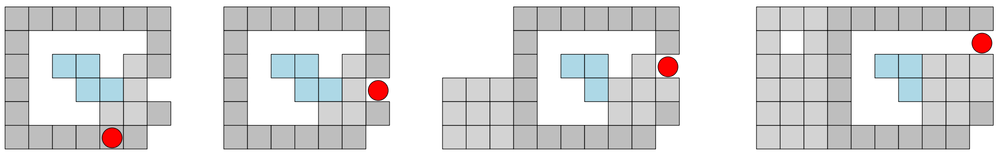

Figure 6 Left: Configuration after the preparation phase. Right three: Different states during phase (2): Scaling an occupied vertex, scaling an empty vertex, and reaching the end of a column.

place a 3×3-square with the middle vertex empty (if a clean up is desired at the end of $\mathcal{A}$, i.e., removing all scaled empty vertices, we fill up the complete row/column with these squares). This guarantees connectivity during the execution and we obtain the following theorem.

▶ **Theorem 3.** *If there is an algorithm $\mathcal{A}$ for some problem Π in the Robots-on-Tiles model with runtime $\mathcal{T}(\mathcal{A})$, such that the robot moves within a $w' \times h'$ rectangle, then there is an algorithm $\mathcal{A}'$ for Π with runtime $O(wh \cdot (c^2 + cw + ch) + \max((w' - w)h', (h' - h)w') + c \cdot \mathcal{T}(\mathcal{A}))$ guaranteeing connectivity during execution.*

5 Conclusion

We demonstrated how geometric algorithms for finite automata can be used to enable very simple robots to perform a number of fundamental but non-trivial construction tasks, such as building a bounding box and scaling a given shape by some constant, that guarantee connectivity between all tiles and robots during their execution.

Future work includes investigation of algorithms without the preceding bounding box construction (e.g. scaling) or distributed algorithms that do not rely on a scaling procedure.

References

1 A. Abdel-Rahman, A. T. Becker, D. E. Biediger, K. C. Cheung, S. P. Fekete, N. A. Gershenfeld, S. Hugo, B. Jenett, P. Keldenich, E. Niehs, C. Rieck, A. Schmidt, C. Scheffer, and M. Yannuzzi. Space ants: Constructing and reconfiguring large-scale structures with finite automata. 2020. Video at `https://www.ibr.cs.tu-bs.de/users/fekete/Videos/Space_submit.mp4`.

2 J. Balanza-Martinez, A. Luchsinger, D. Caballero, R. Reyes, A. A. Cantu, R. Schweller, L. A. Garcia, and T. Wylie. Full tilt: universal constructors for general shapes with uniform external forces. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2689–2708, 2019.

3 A. T. Becker, S. P. Fekete, P. Keldenich, D. Krupke, C. Rieck, C. Scheffer, and A. Schmidt. Tilt assembly: algorithms for micro-factories that build objects with uniform external forces. *Algorithmica*, pages 1–23, 2017.

4 J. J. Daymude, R. Gmyr, K. Hinnenthal, I. Kostitsyna, C. Scheideler, and A. W. Richa. Convex hull formation for programmable matter, 2018.

5 J. J. Daymude, R. Gmyr, A. W. Richa, C. Scheideler, and T. Strothmann. Improved leader election for self-organizing programmable matter. In A. Fernández Anta, T. Jurdzinski, M. A. Mosteiro, and Y. Zhang, editors, *Algorithms for Sensor Systems*, pages 127–140, Cham, 2017. Springer International Publishing.

6 Z. Derakhshandeh, S. Dolev, R. Gmyr, A. W. Richa, C. Scheideler, and T. Strothmann. Brief announcement: Amoebot – a new model for programmable matter. In *Proceedings of the 26th ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '14, pages 220–222, New York, NY, USA, 2014. ACM.

7 Z. Derakhshandeh, R. Gmyr, A. Porter, A. W. Richa, C. Scheideler, and T. Strothmann. On the runtime of universal coating for programmable matter. In Y. Rondelez and D. Woods, editors, *DNA Computing and Molecular Programming*, pages 148–164, Cham, 2016. Springer International Publishing.

8 Z. Derakhshandeh, R. Gmyr, A. W. Richa, C. Scheideler, and T. Strothmann. An algorithmic framework for shape formation problems in self-organizing particle systems. In *Proceedings of the 2nd Annual International Conference on Nanoscale Computing and Communication*, NANOCOM' 15, pages 21:1–21:2, New York, NY, USA, 2015. ACM.

9 Z. Derakhshandeh, R. Gmyr, A. W. Richa, C. Scheideler, and T. Strothmann. Universal coating for programmable matter. *CoRR*, abs/1601.01008, 2016.

10 Z. Derakhshandeh, R. Gmyr, T. Strothmann, R. Bazzi, A. W. Richa, and C. Scheideler. Leader election and shape formation with self-organizing programmable matter. In A. Phillips and P. Yin, editors, *DNA Computing and Molecular Programming*, pages 117–132, Cham, 2015. Springer International Publishing.

11 G. A. Di Luna, P. Flocchini, N. Santoro, G. Viglietta, and Y. Yamauchi. Shape formation by programmable particles. *CoRR*, abs/1705.03538, 2017.

12 S. P. Fekete, R. Gmyr, S. Hugo, P. Keldenich, C. Scheffer, and A. Schmidt. Cadbots: Algorithmic aspects of manipulating programmable matter with finite automata. *CoRR*, abs/1810.06360, 2018.

13 S. P. Fekete, E. Niehs, C. Scheffer, and A. Schmidt. Connected assembly and reconfiguration by finite automata. 2019.

14 R. Gmyr, K. Hinnenthal, I. Kostitsyna, F. Kuhn, D. Rudolph, C. Scheideler, and T. Strothmann. Forming tile shapes with simple robots. In D. Doty and H. Dietz, editors, *DNA Computing and Molecular Programming*, pages 122–138, Cham, 2018. Springer International Publishing.

15 R. Gmyr, I. Kostitsyna, F. Kuhn, C. Scheideler, and T. Strothmann. Forming tile shapes with a single robot. In *33rd European Workshop on Computational Geometry (EuroCG 2017)*, pages 9–12, 2017.

16 C. E. Gregg, J. H. Kim, and K. C. Cheung. Ultra-light and scalable composite lattice materials. *Advanced Engineering Materials*, 20(9):1800213, 2018.

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

18 B. Jenett, D. Cellucci, C. Gregg, and K. Cheung. Meso-scale digital materials: modular, reconfigurable, lattice-based structures. In *ASME 2016 11th International Manufacturing Science and Engineering Conference*. American Society of Mechanical Engineers Digital Collection, 2016.

19 B. Jenett and K. Cheung. Bill-e: Robotic platform for locomotion and manipulation of lightweight space structures. In *25th AIAA/AHS Adaptive Structures Conference*, page 1876, 2017.

20 A. Schmidt, S. Manzoor, L. Huang, A. T. Becker, and S. P. Fekete. Efficient parallel self-assembly under uniform control inputs. *IEEE Robotics and Automation Letters*, 3(4):3521–3528, 2018.