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

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

***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; this is also closely related to the newly emerging field of programmable matter. Aiming for large-scale scalability, both in terms of the number of the cellular components of a structure, as well as the number of robots that are being deployed for construction requires simple yet robust robots and mechanisms, while also dealing with various basic constraints, such as connectivity of a structure during reconfiguration. To this end, we propose an approach that combines ultra-light, cellular building materials with extremely simple robots. We develop basic algorithmic methods that are able to detect and reconfigure arbitrary cellular structures, based on robots that have only constant-sized memory. As a proof of concept, we demonstrate the feasibility of this approach for specific cellular materials and robots that have been developed at NASA.**

I. 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. This is particularly true in space, where the difficulties of expensive supply chains, scarcity of building materials, dramatic costs and consequences of even small errors, and the limitations of outside intervention in case of malfunctions pose a vast array of extreme challenges. Nevertheless, the unquestionable long-term benefits and perspectives of large-scale facilities in space have lead to enormous investments in terms of funds, time, and human capital [3]. As a consequence, the use of advanced methods for autonomous construction in space is indispensable for further progress.

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 [35] that allow the construction of modular, reconfigurable, lattice-based structures [42]; see Figure 1. A second step has been the design of simple autonomous robots [38], [41] that are able to move on the resulting lattice structures and move their elementary cell components, thereby allowing the reconfiguration of the overall edifice; see Figure 2 and Figure 3. Combining such materials and robots in space promises to vastly increase the dimensions of constructible facilities and spacecraft, as well as offering to extend mission capabilities with reconfiguration and re-use [34].

In this paper, we address the next step in this hierarchy: Can we enable extremely simple robots to perform a more complex spectrum of construction tasks for cellular structures in space, such as patrolling and marking the perimeter,

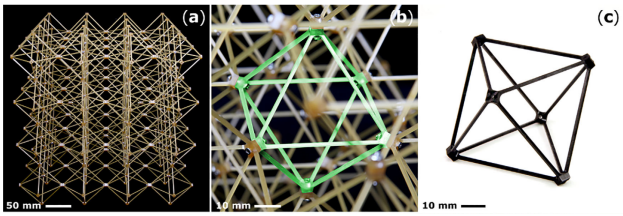

Fig. 1. (a) An assembled cuboctahedral lattice specimen, made from (b) Ultem 2200 (20% glass fiber reinforced polyetherimide) octahedral unit cells (highlighted), termed voxels. (c) A single monolithic RTP 2187 (40% carbon fiber reinforced polyetherimide) injection molded voxel. (See [35].)

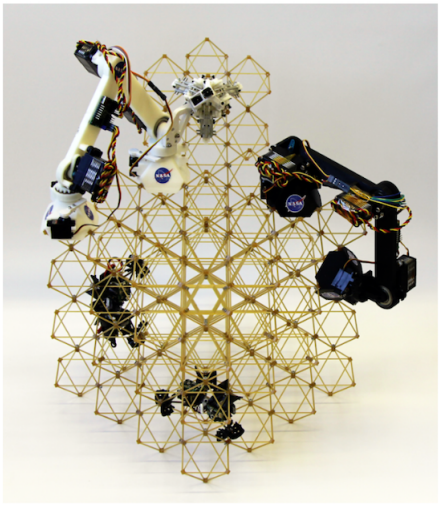

Fig. 2. Modular reconfigurable 3D lattice structure and mobile robots, showing the small size of the robots relative to the structure that they work on, and the parallel use of multiple robots. (See [9].)

* Both authors contributed equally to this document.
[1]Department of Computer Science, TU Braunschweig, Germany. {s.fekete, e.niehs, c.scheffer, arne.schmidt}@tu-bs.de
[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]Center for Bits and Atoms (CBA), Massachusetts Institute of Technology, Cambridge, MA, USA. bej@mit.edu, amira.abdel-rahman@cba.mit.edu

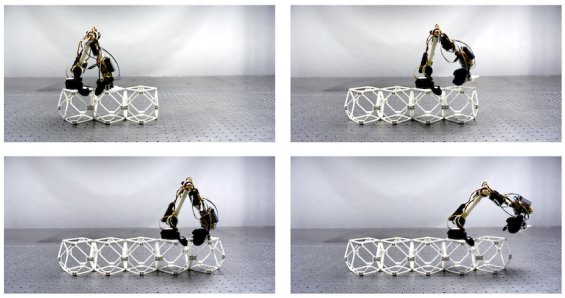

Fig. 3. A BILL-E robot moving on an expanding row of voxels. (See [39].)

scaling up a given seed construction, and a number of other design operations? In the harsh and remote environment of space, relying on powerful CPUs is a serious limitation, not only because of the vulnerability of complex computer hardware, but also because its availability becomes a limiting factor to mass-producing a large number of simple, modular robots in space. As we demonstrate, sophisticated computing hardware is not necessary. Even the extremely limited capabilities of machines with a finite number of states suffice for these tasks. To this end, we provide a suite of algorithmic methods, and demonstrate the feasibility of the resulting approach by implementing it on actual ultra-light material with simple mobile manipulators.

A. Our Results

We present the following results.

1) We show how just two finite-state robots suffice to construct a bounding box for a given connected planar arrangement of grid cells (a *polyomino* P) in a limited number of steps.
2) We provide an algorithmic methods that enables two finite-state robots to construct for a polyomino P with a surrounding bounding box a scaled-up copy of P in a limited number of steps, while preserving connectivity of intermediate arrangements.
3) We also sketch how other basic operations of Computer-Aided Design (such as copying or rotating) can be performed in similar ways.
4) We demonstrate how these methods can be implemented on actual lattice-based structures with simple robots.
5) We provide experimental outcomes for the resulting processing times.

B. Related Work

The structures considered in this work are based on **ultra-light material**, as described by Cheung and Gershenfeld [8] and Gregg et al. [35]. Modular two-dimensional elements mechanically link in 3D to form reversibly assembled composite lattices. This process is not limited by scale, and it enables disassembly and reconfiguration. As shown by Cramer et al. [10] and Jenett et al. [40], large but lightweight structures can be built from these components. Jenett et al. have developed autonomous robots that move on the surface [38], [39] or within the cellular structure [41]. With the help of these robots, individual cells can be attached to an existing assembly, or moved to a different location. An approach for global optimization of a corresponding motion plan has been described by Costa et al. [9], while the design of hierarchical structures was addressed by Jenett et al. [43].

Assembly by simple robots has also been considered at the micro scale, where global control control is used for supplying the necessary force for moving agents, e.g., see Becker et al. [2] for the corresponding problem of motion planning, Schmidt et al. [48] for using this model for assembling structures, and Balanza-Martinez et al. [1] for theoretical characterizations. Distributed self-assembly for modular robots with limited computing resources was studied by Tucci et al. [49]. Self-configuration of robots themselves has been considered by Naz et al. [44]. A basic model in which robots themselves are used as building material was introduced by Derakhshandeh et al. [16], [17]. This resembles Claytronics robots like Catoms, see Goldstein and Mowry [33].

A large spectrum of methods for **tile-based assembly in biology** has also been considered. Examples include DNA self-assembly, introduced by Winfree [50], [51] and extended to a variety of models [7], [13]. See Patitz [47] for a survey.

On the algorithmic side, there has been a considerable amount of work dealing with **robots or agents on graphs**. Blum and Kozen [5] showed that two finite automata can jointly search any unknown maze. Other work has focused on exploring general graphs (e.g., [24], [27], [46]), as a distributed or collaborative problem using multiple agents (e.g. [4], [6], [11], [25]) or with space limitations (e.g. [21], [26]–[29]).

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 [14] and later a generalized variant, the general Amoebot model [19]. 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. With just a few possible movements, these particles can be formed into different shapes like lines, triangles or hexagons [16], and a leader out of all particles can be elected [12], [19]. A universal shape formation algorithm within the Amoebot model was described by Di Luna et al. in [20]. An algorithm for solving the problem of coating an arbitrarily shaped object with a layer of self-organizing programmable matter was presented in [18] and analyzed in [15]. Other models with active particles were introduced in [52] as the Nubot model and in [37] with modular robots. In [30], Gmyr et al. introduced a model with two types of particles: active robots acting like a deterministic finite automaton and passive tile particles. Furthermore, they presented algorithms for shape formation [32] and shape recognition [31] using robots on tiles. Using the same "Robot-on-Tiles" model as we do in our work, Hugo presented algorithms for recognizing convexity of a given polyomino and counting the number of tiles or

corners in her Bachelor thesis [36]. Finally, Fekete et al. 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 in [22].

II. Preliminaries

In the following, we introduce models, general definitions as well as a description of the underlying limitations.

A. Model

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 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 adjacent to an empty vertex. For simplicity we show the boundary as a red line around P (see also Figure 4 (a)). 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 axis-aligned rectangle enclosing but not touching P; it will be denoted by $bb(P)$ (see Fig. 5). Because the bounding box and polyomino are comprised of identical tiles, a gap is necessary to differentiate the two.

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 case of multiple robots (Figure 6 (b)), 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.

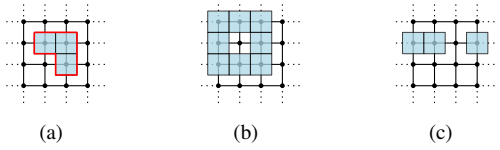

Fig. 4. (a) The red line indicates the boundary of P, denoted by ∂P. (b) A non-simple polyomino with one hole. (c) The tiles on the grid induce two separate connected components.

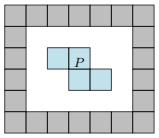

Fig. 5. A polyomino P and its gray colored bounding box surrounding P.

Connectivity in the sense of this work is ensured if the union of all placed tiles and all used robots is completely connected. Accordingly, a robot can hold two components together, e.g., as shown in Figure 6 (c).

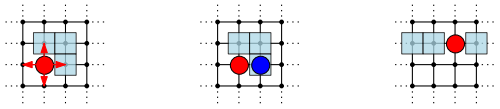

Fig. 6. (a) One robot and its possible moves. (b) Two robots on the grid. (c) Robots can hold separate connected components together.

III. Constructing a Bounding Box

Fekete et al. [22] showed how to construct a bounding box around a polyomino. However, that algorithm does not guarantee connectivity. We describe an algorithm to construct the bounding box keeping connectivity after each step. Due to space constraints, we only sketch technical details; see the full version of the paper [23] 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 (marked red) is the leader.

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 and moving downwards whenever possible. The search is done by the leader robot R_1, followed by R_2. Then R_2 positions itself on the first vertex beneath this locally y-minimal vertex. 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. In the latter case 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 Fig. 7). 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 a lower y-coordinate than the current starting position. Therefore,

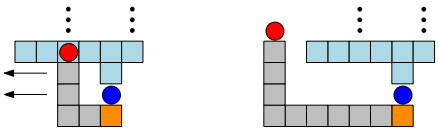

Fig. 7. (a) R_1 hits a tile belonging to P. (b) The triggered shifting process is finished.

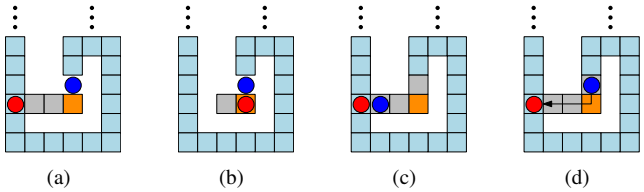

Fig. 8. Traversing a gap by building a bridge

we build a bridge to traverse this gap, as shown in Fig. 8. Afterwards, we can restart from phase (1).

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.

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 9(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 9(b)-(d)). 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 [22]; see [23] for full details.

If we know in advance that the given polyomino contains no holes, then we can build a non-simple bounding box. This requires only one robot, because we can at any moment distinguish the polyomino from the bounding box without having a second robot holding both parts together. This yields the following corollary.

Corollary 1: Given a simple 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 one robot in $O(\max(w,h)\cdot wh)$ steps.

IV. Scaling Polyominoes

Now we consider scaling a given shape by a factor c. Note that reducing the size of a polyomino by a factor ("scaling down") can then be done in a similar fashion. In the following we assume that the robot R_1 already built the bounding box and is positioned on one of its tiles.

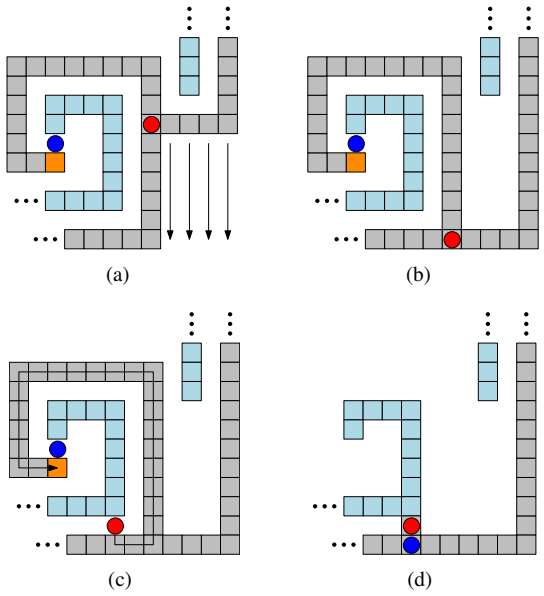

Fig. 9. The second case of finishing the bounding box. (a) An already constructed part of the bounding box is hit. (b) The last boundary side is shifted. (c) R_1 found a suitable new connectivity vertex above the southern side, places a tile and retraces its path to the initial starting position. (d) The unnecessary part of the bounding box is removed and both robots catch up to the new connection.

A. 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, i.e., rightmost 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 Fig. 10(a)). 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.

In phase (2), we simply search for the tile to scale, and place the row marker one vertex upwards. For possible cases, see Fig. 10. 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 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. Otherwise, we do not move upwards and build the square after the leftward moves. If the vertex to scale is empty, then we leave out one tile within the square.

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

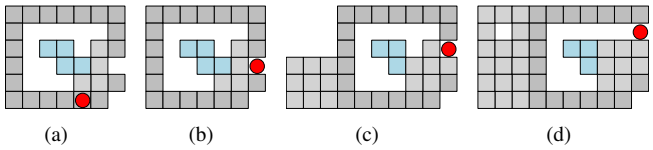

Fig. 10. (a) Configuration after the preparation phase. (b)-(d) Cases that appear during the scaling: (b) Scaling an occupied vertex; (c) scaling an empty vertex; (d) reaching the end of a column.

right to left. If necessary, all scaled empty tiles can also be removed by one scan through the scaled field.

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.

Proof: **Correctness:** We scan through the whole bounding box of P and scale every position. This implies that we scale every tile of P. Because we scale columnwise, we ensure that every scaled tile is built at the correct position. Connectivity is guaranteed because we never remove a tile that is necessary to have connectivity.

Time: Each of the $w \cdot h$ vertices within the bounding box of P is scaled. To this end, the robot has to move $O(c(w+h))$ steps to reach the position, where the scaled vertex needs to be constructed. A further $O(c^2)$ steps are needed to construct the tile. Finding the next vertex to scale takes $O(c(w+h))$ steps, including going back, find the correct column and row, and moving the row and column marker. In total we have a runtime of $O(wh(c^2 + cw + ch))$. ■

B. Adapting Algorithms

As shown in Fig. 11, 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 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.

V. Simulation

As a key step in realizing a full robotic implementation of the bounding box construction, a tile-based simulator was developed[1]. This python-based simulation allows for a clearer understanding of the subtle details in the execution of the approach. It models the states of the tiles as well as the robots and elucidates the state transitions. The visualization provides a snapshot of these states and how they unfold. It aides in understanding the scalability of the approach as well as the behavior of the approach on different shapes. It also provided the motion plan for the full 3D simulation of the algorithm with the virtual BILL-E robot.

[1] https://github.com/AlienHunterD/2DTileRobot

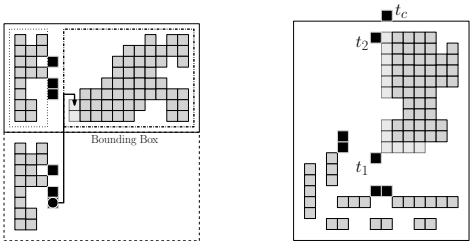

Fig. 11. Figures from [22] showing how to copy (left) or rotate (right) a given shape. We can clearly see that the tiles are not connected. Theorem 3 guarantees that this kind of construction can be modified to be performed in a connected fashion.

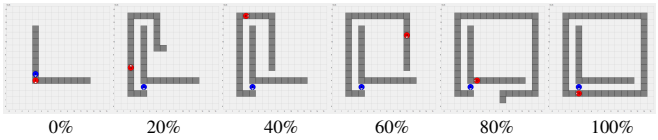

Fig. 12. 2D simulation snapshots from building a bounding box for an L-shaped polyomino. See video [45] for animation.

The implemented state machine has 34 states, which include some optimizations for shifting long stretches of tiles. When the robots are in close proximity, they behave synchronously. In each state the robots first sense one unit in the directions left, up, right, and down. Each step in the simulation consists of one or all of the following actions: {look; communicate; turn; move one space; add, move or delete a tile; turn}. A build sequence from the simulator is shown in Fig. 12.

A plot showing the percentage of steps spent in four classes of states is shown in Fig. 14, for constructing bounding boxes around L-shaped polyominoes. The state classes for these plots are *Initial Search* (which searches for the local y-minimum of the seed polyomino), *Add/Shift Tiles* (which either adds a tile or shifts an already placed tile by 90°), *Delete Tiles* (which removes a tile), or *Move/Search* (which either moves to the end of the bounding box under construction or searches to see if the current tile is part of the bounding box or the seed polyomino).

As the size of the polyomino grows, the number of steps required grows quadratically. The time required to *Move/Search* dominates the other classes. The number of steps required to construct a bounding box around six simple shapes of polyominoes are compared in Fig. 15. Filled squares require the fewest steps to construct a bounding box, while L-shaped sets require considerably more time. Most of this extra time is spent searching to determine if an encountered tile is part of the polyomino or bounding box. Four classes of shapes that are identical up to a rotation are

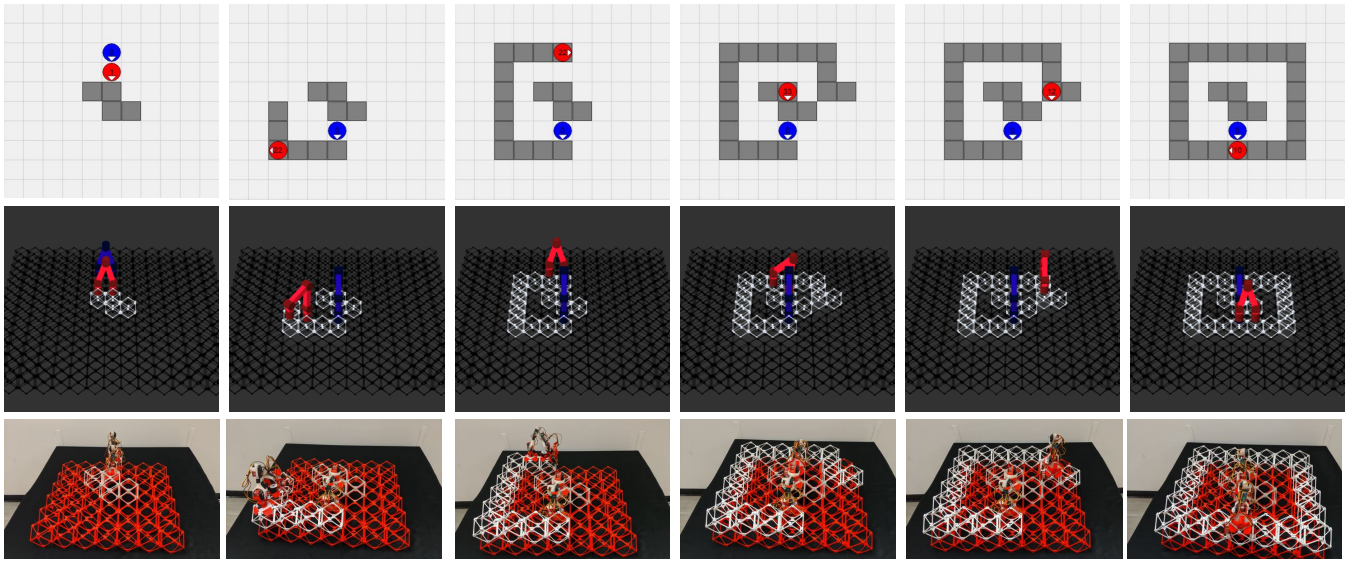

Fig. 13. Snapshots from building a bounding box for z-shaped polyomino using 2D simulator, 3D simulator, and staged hardware robots, synchronized so all are shown at steps {0, 24, 48, 72, 96, 120}. See video for animation: `https://youtu.be/K80sV5Xf7v4` [45].

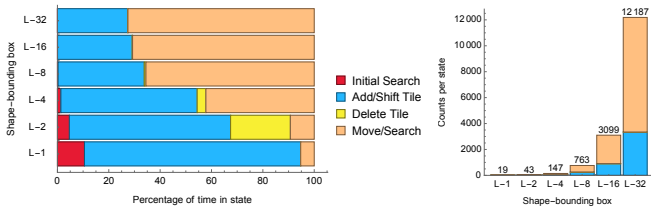

Fig. 14. Percentage of time spent in states for an L-shaped seed polyomino. The L-1 is a single tile, while the L-32 extends from the bottom-left tile a single row of tiles 32 wide and a single column 32 tiles tall.

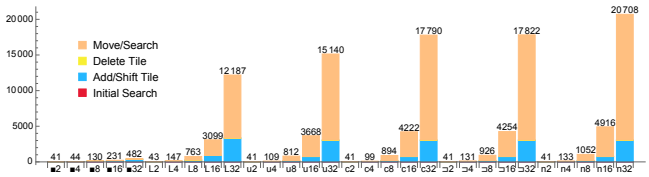

Fig. 15. Steps required for six canonical shapes. All shapes fill an $n \times n$ bounding box, with $n \in \{2, 4, 8, 16, 32\}$. Shapes include filled squares, and five shapes composed of single width polyomino lines: L-shapes, u-shapes, c-shapes, ⊐-shapes, and n-shapes. The time requirements increase from left to right.

u, c, ⊐, and n-shaped sets. The differences in time for these classes reflect the arbitrary choice to turn clockwise when determining if a tile belongs to the polyomino. These simulations show many opportunities for improved efficiency. The time for L-shapes is dominated by running the routine to determine if the tile encountered is the boundary or the seed polyomino. Adding more states could approximate wall following.

The tile-based simulator was used to construct a 3D simulation using the BILL-E robots and magnetic voxels. The tile-based simulator, the 3D simulator, and two hardware robots constructing a bounding box are shown in Fig. 13. Moving from the tile-based approach required modifications. Among those are: (1) The robot must physically reach out to look at a neighboring voxel (and thus must not collide with another robot). (2) The robot can only move from one tile to the neighbor by performing a "cartwheel" move. In practice, performing an inch-worm maneuver that places a leg two tiles away, then moves the trailing leg forward, moves the robot more quickly. (3) At this point, the implementation details of obtaining a voxel to add or deleting a voxel are omitted. In the future, robots may carry a supply of voxels, move them to or from appropriately placed depots when this supply is exhausted, be supplied by a number of dedicated gopher robots, or may even construct and consume voxels (i.e. 3D printing/ filament recycling).

VI. 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, and also provided a practical realization.

There is a whole range of possible extensions. Is it possible to scale general polyominoes without the preceding bounding box construction? A possible approach could be to cut the polyomino into a subset of monotone polyominoes, which could be handled separately. Expanding the existing repertoire of operations to three-dimensional configurations and operations is another logical step. An equally relevant challenge is to develop distributed algorithms with multiple robots that are capable of solving a range of problems with the requirement of connectivity, without having to rely on the preceding scaling procedure that we used in our work. Other questions arise from additional requirements of real-world applications, such as the construction and reconfiguration of space habitats.

REFERENCES

[1] 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 *30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2019, pp. 2689–2708.

[2] 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*, pp. 1–23, 2017.

[3] W. K. Belvin, W. R. Doggett, J. J. Watson, J. T. Dorsey, J. E. Warren, T. C. Jones, E. E. Komendera, T. Mann, and L. M. Bowman, "In-space structural assembly: Applications and technology," in *3rd AIAA Spacecraft Structures Conference*, 2016, p. 2163.

[4] M. A. Bender and D. K. Slonim, "The power of team exploration: two robots can learn unlabeled directed graphs," in *35th Annual Symposium on Foundations of Computer Science (FOCS)*, 1994, pp. 75–85. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=365703

[5] M. Blum and D. Kozen, "On the power of the compass (or, why mazes are easier to search than graphs)," in *19th Annual Symposium on Foundations of Computer Science (FOCS)*, 1978, pp. 132–142. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4567972

[6] P. Brass, F. Cabrera-Mora, A. Gasparri, and J. Xiao, "Multirobot tree and graph exploration," *IEEE Transactions on Robotics*, vol. 27, no. 4, pp. 707–717, Aug 2011.

[7] C. Chalk, E. Martinez, R. Schweller, L. Vega, A. Winslow, and T. Wylie, "Optimal staged self-assembly of general shapes," *Algorithmica*, vol. 80, no. 4, pp. 1383–1409, 2018.

[8] K. C. Cheung and N. Gershenfeld, "Reversibly assembled cellular composite materials," *Science*, vol. 341, no. 6151, pp. 1219–1221, 2013.

[9] A. Costa, A. Abdel-Rahman, B. Jenett, N. Gershenfeld, I. Kostitsyna, and K. Cheung, "Algorithmic approaches to reconfigurable assembly systems," in *IEEE Aerospace Conference*, 2019, pp. 1–8.

[10] N. B. Cramer, D. W. Cellucci, O. B. Formoso, C. E. Gregg, B. E. Jenett, J. H. Kim, M. Lendraitis, S. S. Swei, G. T. Trinh, K. V. Trinh *et al.*, "Elastic shape morphing of ultralight structures by programmable assembly," *Smart Materials and Structures*, vol. 28, no. 5, p. 055006, 2019.

[11] S. Das, P. Flocchini, S. Kutten, A. Nayak, and N. Santoro, "Map construction of unknown graphs by multiple agents," *Theoretical Computer Science*, vol. 385, no. 1, pp. 34 – 48, 2007.

[12] J. J. Daymude, R. Gmyr, A. W. Richa, C. Scheideler, and T. Strothmann, "Improved leader election for self-organizing programmable matter," in *Algorithms for Sensor Systems*, Cham, 2017, pp. 127–140.

[13] E. D. Demaine, M. L. Demaine, S. P. Fekete, M. Ishaque, E. Rafalin, R. T. Schweller, and D. L. Souvaine, "Staged self-assembly: nanomanufacture of arbitrary shapes with o (1) glues," *Natural Computing*, vol. 7, no. 3, pp. 347–370, 2008.

[14] Z. Derakhshandeh, S. Dolev, R. Gmyr, A. W. Richa, C. Scheideler, and T. Strothmann, "Brief announcement: Amoebot – a new model for programmable matter," in *26th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2014, pp. 220–222.

[15] Z. Derakhshandeh, R. Gmyr, A. Porter, A. W. Richa, C. Scheideler, and T. Strothmann, "On the runtime of universal coating for programmable matter," in *22nd International Conference on DNA Computing and Molecular Programming (DNA)*, 2016, pp. 148–164.

[16] 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 *2nd International Conference on Nanoscale Computing and Communication (NANOCOM)*, 2015, pp. 21:1–21:2.

[17] ——, "Universal shape formation for programmable matter," in *Proceedings of the 28th ACM Symposium on Parallelism in Algorithms and Architectures*, 2016, pp. 289–299.

[18] ——, "Universal coating for programmable matter," *Theoretical Computer Science*, vol. 671, pp. 56–68, 2017.

[19] 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 *21st International Conference on DNA Computing and Molecular Programming (DNA)*, 2015, pp. 117–132.

[20] G. A. Di Luna, P. Flocchini, N. Santoro, G. Viglietta, and Y. Yamauchi, "Shape formation by programmable particles," *Distributed Computing*, vol. 33, no. 1, pp. 69–101, 2020.

[21] K. Diks, P. Fraigniaud, E. Kranakis, and A. Pelc, "Tree exploration with little memory," *Journal of Algorithms*, vol. 51, no. 1, pp. 38 – 63, 2004.

[22] 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*, vol. abs/1810.06360, 2018. [Online]. Available: https://arxiv.org/pdf/1810.06360.pdf

[23] S. P. Fekete, E. Niehs, C. Scheffer, and A. Schmidt, "Connected assembly and reconfiguration by finite automata," *CoRR*, 2019. [Online]. Available: https://arxiv.org/pdf/1909.03880.pdf

[24] R. Fleischer and G. Trippen, "Exploring an unknown graph efficiently," in *13th European Symposium on Algorithms (ESA)*, 2005, pp. 11–22.

[25] P. Fraigniaud, L. Gasieniec, D. R. Kowalski, and A. Pelc, "Collective tree exploration," *Networks*, vol. 48, no. 3, pp. 166–177, 2006.

[26] P. Fraigniaud and D. Ilcinkas, "Digraphs exploration with little memory," in *21st Symposium on Theoretical Aspects of Computer Science (STACS)*, 2004, pp. 246–257.

[27] P. Fraigniaud, D. Ilcinkas, G. Peer, A. Pelc, and D. Peleg, "Graph Exploration by a Finite Automaton," *Theoretical Computer Science*, vol. 345, no. 2-3, pp. 331–344, Nov. 2005.

[28] L. Gasieniec, A. Pelc, T. Radzik, and X. Zhang, "Tree exploration with logarithmic memory," in *18th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2007, pp. 585–594.

[29] L. Gasieniec and T. Radzik, "Memory efficient anonymous graph exploration," in *34th Workshop Graph-Theoretic Concepts in Computer Science (WG)*, 2008, pp. 14–29.

[30] 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, pp. 9–12.

[31] R. Gmyr, K. Hinnenthal, I. Kostitsyna, F. Kuhn, D. Rudolph, and C. Scheideler, "Shape Recognition by a Finite Automaton Robot," in *43rd International Symposium on Mathematical Foundations of Computer Science (MFCS)*, vol. 117, 2018, pp. 52:1–52:15.

[32] R. Gmyr, K. Hinnenthal, I. Kostitsyna, F. Kuhn, D. Rudolph, C. Scheideler, and T. Strothmann, "Forming tile shapes with simple robots," in *24th International Conference on DNA Computing and Molecular Programming (DNA)*, 2018, pp. 122–138.

[33] S. C. Goldstein and T. Mowry, "Claytronics: A scalable basis for future robots," *Robosphere, Nov*, 2004.

[34] C. E. Gregg, B. Jenett, and K. C. Cheung, "Assembled, modular hardware architectures - what price reconfigurability?" in *2019 IEEE Aerospace Conference*, 2019, pp. 1–10.

[35] C. E. Gregg, J. H. Kim, and K. C. Cheung, "Ultra-light and scalable composite lattice materials," *Advanced Engineering Materials*, vol. 20, no. 9, p. 1800213, 2018.

[36] S. Hugo, "Robots on tiles: Recognition of polyomino properties using constant memory," Bachelor's Thesis, TU Braunschweig, 2018.

[37] F. Hurtado, E. Molina, S. Ramaswami, and V. Sacristán, "Distributed reconfiguration of 2D lattice-based modular robotic systems," *Autonomous Robots*, vol. 38, no. 4, pp. 383–413, Apr 2015.

[38] B. Jenett and K. Cheung, "Bill-e: Robotic platform for locomotion and manipulation of lightweight space structures," in *25th AIAA/AHS Adaptive Structures Conference*, 2017, p. 1876.

[39] B. Jenett, A. Abdel-Rahman, K. C. Cheung, and N. Gershenfeld, "Material-robot system for assembly of discrete cellular structures," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4019–4026, 2019.

[40] B. Jenett, S. Calisch, D. Cellucci, N. Cramer, N. Gershenfeld, S. Swei, and K. C. Cheung, "Digital morphing wing: active wing shaping concept using composite lattice-based cellular structures," *Soft Robotics*, vol. 4, no. 1, pp. 33–48, 2017.

[41] 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)*, 2017, pp. 5474–5479.

[42] 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*, 2016.

[43] B. Jenett, C. Gregg, D. Cellucci, and K. Cheung, "Design of multifunctional hierarchical space structures," in *IEEE Aerospace Conference*, 2017, pp. 1–10.

[44] A. Naz, B. Piranda, J. Bourgeois, and S. C. Goldstein, "A distributed self-reconfiguration algorithm for cylindrical lattice-based modular robots," in *IEEE 15th International Symposium on Network Computing and Applications (NCA)*, 2016, pp. 254–263.

[45] E. Niehs, A. Schmidt, C. Scheffer, D. E. Biediger, M. Yannuzzi, B. Jenett, A. Abdel-Rahman, K. C. Cheung, A. T. Becker, and S. P. Fekete, "Video: Recognition and reconfiguration of lattice-based cellular structures by simple robots," March 2020. [Online]. Available: https://youtu.be/K80sV5Xf7v4

[46] P. Panaite and A. Pelc, "Exploring unknown undirected graphs," *Journal of Algorithms*, vol. 33, no. 2, pp. 281 – 295, 1999.

[47] M. J. Patitz, "An introduction to tile-based self-assembly and a survey of recent results," *Natural Computing*, vol. 13, no. 2, pp. 195–224, 2014.

[48] 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*, vol. 3, no. 4, pp. 3521–3528, 2018.

[49] T. Tucci, B. Piranda, and J. Bourgeois, "A distributed self-assembly planning algorithm for modular robots," in *17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2018, pp. 550–558.

[50] E. Winfree, "Algorithmic self-assembly of DNA," Ph.D. dissertation, California Institute of Technology, 1998.

[51] E. Winfree, F. Liu, L. A. Wenzler, and N. C. Seeman, "Design and self-assembly of two-dimensional DNA crystals," *Nature*, vol. 394, no. 6693, p. 539, 1998.

[52] D. Woods, H.-L. Chen, S. Goodfriend, N. Dabby, E. Winfree, and P. Yin, "Active self-assembly of algorithmic shapes and patterns in polylogarithmic time," in *4th Conference on Innovations in Theoretical Computer Science (IITCS)*, 2013, pp. 353–354.