All for One and One for All: An $O(1)$-Musketeers Universal Transformation for Rotating Robots

Matthew Connor
Department of Computer Science, University of Liverpool, UK

Othon Michail
Department of Computer Science, University of Liverpool, UK

George Skretas
Hasso Plattner Institute, University of Potsdam, Germany

Abstract

In this paper, we settle the main open question of [Michail, Skretas, Spirakis, ICALP'17], asking what is the family of two-dimensional geometric shapes that can be transformed into each other by a sequence of rotation operations, none of which disconnects the shape. The model represents programmable matter systems consisting of interconnected modules that perform the minimal mechanical operation of 90° rotations around each other. The goal is to transform an initial shape of modules A into a target shape B. Under the necessary assumptions that the given shapes are connected and have identical colourings on a checkered colouring of the grid, and using a seed of only constant size, we prove that any pair of such shapes can be transformed into each other within an optimal $O(n^2)$ rotation operations none of which disconnects the shape.

2012 ACM Subject Classification Theory of computation → Computational geometry; Theory of computation → Design and analysis of algorithms

Keywords and phrases programmable matter, universal transformation, reconfigurable robotics, shape formation, centralised algorithms

Digital Object Identifier 10.4230/LIPIcs.SAND.2024.9

1 Introduction

Programmable matter refers to matter that can change its physical properties algorithmically. It is envisioned as a collection of modules connected to each other to form a *shape*. Due to size and other constraints of the individual modules, limited actuation and sensing capabilities are available to them, which a program uses to enable the interaction of the material with its surroundings and to control its structural dynamics. The relevant theoretical literature has almost exclusively focused on designing algorithms (either centralised or distributed) for the task of transforming a given initial shape A into a given target shape B and characterising the families of shapes that can be transformed into each other within a given programmable matter model. Transformations should additionally be efficient, which for sequential transformations is measured by the total number of individual actuation operations.

In [20] (and its journal version [21]), Michail et al. studied a model of programmable matter in which modules are represented by nodes drawn within the cells of a two-dimensional square grid. Nodes are connected to any nodes orthogonally adjacent to them (their *neighbours*) to form a shape. The collection of nodes can be reconfigured between shapes through minimal types of movements. One of the considered movements was *rotation*: a node can rotate 90° around a neighbour provided that the rotating node's trajectory is free from other nodes. The authors introduced the problem of characterising which families of connected shapes can be transformed into each other via rotation movements. If global connectivity need not be preserved, they proved that the a decision version of the problem (called Rot-

TRANSFORMABILITY) is in $\mathbf{P}$[1]. They proved this through a constructive exact characterisation of the shapes that can be transformed into each other. For the ROTC-TRANSFORMABILITY version of the problem, in which global connectivity must be preserved after every movement, they proved inclusion in **PSPACE** and highlighted that surprisingly small *seeds* can enable transformations that are otherwise infeasible. The main problem they left open was asking if there exists a universal centralised transformation for ROTC-TRANSFORMABILITY under the assumption of a constant-size seed.

A more general version of the model is one that combines rotation and sliding, where a node can additionally slide over pairs of consecutive nodes. For this version, Dumitrescu et al. [14] had studied distributed transformations and had conjectured that universal connectivity-preserving transformation is possible. This was proven to be correct in [13] and independently in [20]. Both the rotation and the combined rotation and sliding models aim to represent minimal and, thus, cost- and energy-efficient mechanical operations and to show that real implementations of programmable matter using them could hope to have universal reconfiguration capabilities. The focus on minimal operations is also justified by the engineering objective to minimise the size of individual modules in order to improve the granularity of the material without sacrificing its global actuation capabilities.

However, with rotation alone not all pairs of connected shapes of the same number of nodes can be transformed into each other. For example, there are shapes, like a rhombus, that are completely blocked and other shapes, like a line, that are blocked within a small final strongly connected component of the shape-reachability graph if connectivity is to be preserved. Because of this, universal transformation by rotation cannot be achieved without additional assumptions. Such an assumption, introduced in [20], is to use a small additional set of nodes, called a *seed* (or d-seed, where d is the number of nodes of the seed), that when placed appropriately on the perimeter of the shape can trigger the otherwise infeasible transformation. The question posed was: *Is there a reasonably small seed (i.e., of constant size), whose availability enables universal connectivity-preserving transformation when the only available movement is rotation?*

Direct progress on this open question was made by Connor et al. [6] and Connor and Michail [5]. In [6], a 4-seed was shown to be sufficient for solving the problem on a restricted family of shapes, called *nice shapes*. These were first defined in [3] as all shapes S having a central line L, where, for all nodes $u \in S$, either $u \in L$ or u is connected to L by a line of nodes perpendicular to L. The first breakthrough towards universality was achieved in [5], where the problem was shown to be solvable for all orthogonally convex shapes by using a minimal 3-seed [2]. The family of nice shapes and that of orthogonally convex shapes are not directly comparable as each contains at least one shape that does not belong to the other. Nevertheless, orthogonally convex shapes have appeared to be much richer in structure and harder to transform. We extend the techniques developed in [5] to obtain an $O(1)$-seed universal transformation, that is, one that works for all pairs of connected *colour-consistent* shapes. Two shapes are colour-consistent if they have identical colourings on a checkered colouring of the grid: as a node cannot change colour by rotating, any shapes that can be transformed into each other must be colour-consistent.

---

[1] Polynomial time in this context refers to the worst-case time complexity of an algorithm that decides if two given shapes A and B can be transformed into each other. It should not be confused with the efficiency of a transformation between the shapes, which is measured in total movements for sequential transformations and in time-steps of parallel movements for parallel transformations.

[2] A shape S is *orthogonally convex* if for any two nodes u, v in a horizontal or vertical line of the grid, all cells between u and v are occupied by S.

There is a lower bound on the number of worst-case movements required by a transformation, which is quadratic in the number of nodes. It is based on a measure of "distance" between the initial and the target shape, and applies to all models in which every movement reduces the total distance by at most a constant, also affecting solutions that do not preserve connectivity. Because of this, optimal sequential transformations perform $O(n^2)$ movements.

In another model which bares some similarities to the rotation and the combined rotation and sliding models, Akitaya et al. [1] studied a different type of movement, called *pivoting*. Pivoting allows a square-shaped node to emulate sliding and rotation. Both these operations happen by fixing an arm on a shared corner between two squares and rotating one of the squares along that arm. The rotation movement we consider here is not directly comparable to pivoting. Pivoting requires more empty space around the moving node than rotation. On the other hand, it can "slide" a node to an orthogonally adjacent cell, thus, in contrast to what holds for rotation, pivoting nodes have no *a priori* unreachable locations. Akitaya et al. accomplished universal transformation in $O(n^2)$ pivoting movements using a "bridging" procedure assisted by at most 5 seed-nodes, which they called *musketeers*.

2 Contribution and Approach

We study RotC-Transformability, the problem of characterising the families of connected shapes that can be transformed into each other via rotation movements without breaking connectivity. As our focus is on the feasibility and complexity of transformations, our approach is naturally based on structural characterisations and centralised procedures. Structural and algorithmic progress is expected to facilitate more applied future developments, such as distributed implementations.

When rotation is combined with sliding, the algorithmic strategy to establish universality [13, 20] is quite intuitive. The goal is to show that any two connected shapes of the same number of nodes can be transformed into each other. Due to reversibility of these movements, it is sufficient to show that any connected shape S of n nodes can be transformed into a straight line of length n. The line is the *canonical shape* of this strategy and all its transformations will go through it. The strategy is based on the observation that, by combining rotation and sliding, a node can traverse the perimeter of S. To transform S into a line, a position on the perimeter of S from which the line can grow to its full length is fixed. It can then be shown that there is always a node to remove from the perimeter without disconnecting the shape. The algorithm moves the node along the perimeter until it reaches the line and places it in the empty cell adjacent to the furthest endpoint of the line. It then repeats by removing another node from the perimeter.

Rotation alone is also quite powerful if connectivity need not be preserved. As there is an infinite family of shapes which are completely blocked under rotation (the rhombi), one cannot hope to achieve universality. Nevertheless, there is a strategy that works for all the remaining shapes [20]. The canonical shape of this strategy is the *line-with-leaves*, a family of shapes with maximal colour capacity. The transformation removes nodes from the shape in pairs and transports them to the line-with-leaves, which can be constructed anywhere on the grid, not necessarily being connected to the original shape.

When connectivity must be preserved, transformations by rotation alone can be notoriously difficult. There are two main sources of this difficulty. We still cannot hope to achieve universal transformation free from additional assumptions due to blocked shapes, whose class is now increased by the requirement to preserve connectivity. Moreover, as it cannot change colour in a checkered colouring of the grid, a node cannot in general traverse the perimeter of a shape. It was known from [20] that this remains impossible for up to 4 nodes working together: 4 or less nodes cannot traverse the perimeter of a straight line.

The transformations of [6] and [5] are based on the approach of using a small set of nodes (called a *robot*) that work together to traverse the perimeter of the shape and transport other nodes, thus simulating the single-node traversal of the combined rotation and sliding model. The robot of [6] consisted of 4 nodes and the robot of [5] of 6 nodes. The reason that the 4-robot of [6] does not violate the lower bound of [20] on the number of nodes needed to traverse the perimeter, is that, due to their special structure, nice shapes can be transformed through partial traversals of their perimeter. Both papers used seeds of 4 and 3 nodes, respectively, to enable the initial formation of the robot and deal with blocked shapes.

The result of [5] is that, under the assumption of a 3-seed, any pair of colour-consistent orthogonally convex shapes can be transformed into each other. The 3-seed is optimal: there are blocked shapes for which non-trivial transformations cannot be enabled by a smaller seed. The transformation uses the 3-seed to remove a 6-robot. It proceeds in phases to transform the given shape into a canonical shape, which is an orthogonally convex variant of the line-with-leaves. In each phase, the 6-robot removes the next node from the perimeter according to an elimination sequence, transports it around the perimeter until it reaches the canonical shape, and places the node in the next available cell of the canonical shape.

A key difficulty in generalising the approach of [5] to any shape is that the perimeter of an arbitrary shape can be a lot harder to traverse. Though, as in [5], there is always a node on the perimeter that can be removed, a robot of nodes might not have enough space to reach that node and it is not even clear if it can successfully traverse the perimeter with or without carrying the node. For example, all removable nodes might be concealed within pockets formed by the perimeter that are too narrow for the robot to access and there can be concave parts of the perimeter to which the traversal of [5] does not readily transfer. As a result, both the elimination sequence and the robot traversal must be carefully redesigned.

We overcome these difficulties and show how to transform any shape S into a variant of a line-with-leaves by a sequence of $O(n^2)$ rotations. Reversibility of rotation then implies a universal transformation between pairs of shapes going through the line-with-leaves. We first argue that there is a placement of an $O(1)$-seed on S from which a "good" initial configuration for the transformation can be obtained, having a 6-robot and an adequate initialisation of the line-with-leaves on the *reachable* part of the perimeter of S. We show how to compute an elimination sequence of the nodes of S that guarantees a generation sequence of the line-with-leaves that never exceed its colour capacity. As long as there are reachable nodes to be removed as required by the elimination sequence, the 6-robot picks a reachable node, transports (as a 7-robot) the node to the line-with-leaves, and places the node in an appropriate cell adjacent to the line-with-leaves, as specified by the corresponding generation sequence. This entails showing that both a 6-robot and a 7-robot can traverse the whole reachable perimeter of S, by traversing reachable parts and bypassing unreachable parts. If there are no reachable nodes to be removed, we show how closing a "bottleneck lid" and compressing the shape (if needed) allows the elimination sequence to keep making progress.

In Section 3, we discuss other related work. In Section 4, we formally define the model used in this paper. Section 5 presents the universal transformation. In Section 6, we conclude and state some open problems.

3 Other Related Work

As the development of these systems continues, it becomes increasingly necessary to develop theoretical models which are capable of describing and explaining the emergent properties, possibilities and limitations of such systems in an abstract and fundamental manner. To this

end, models have been developed for programmable matter. For example, algorithmic self-assembly [11, 23, 24] focuses on programming molecules like DNA to grow in a controllable way, and the Abstract Tile Assembly Model [25, 30], the Kilobot model [26], the Robot Pebbles system [17], and the nubot model [31], have all been developed for this area. Network Constructors [22] is an extension of population protocols [4] that allows for network formation and reconfiguration. The latter model is formally equivalent to a restricted version of chemical reaction networks, which "are widely used to describe information processing occurring in natural cellular regulatory networks" [27, 12]. The CATOMS system [28, 29, 15] is a further implementation which constructs 3D shapes by first creating a "scaffolding structure" as a basis for construction. Finally, there is extensive research into the amoebot model [8, 7, 10, 9], where finite automata on a triangular lattice follow a distributed algorithm to achieve a desired goal, including a recent extension [16] to a circuit-based model.

Almalki and Michail [2], building on the insertion operations of [31] and the growth processes on graphs by Mertzios et al. [19], investigated what families of shapes can be grown in time polylogarithmic in their size by using only growth operations.

4 Model

We consider the case of programmable matter on a two-dimensional square grid, with each cell of the grid being uniquely referred to by its (x, y) coordinates. Such a system consists of a set V of n nodes. Each node is viewed as a spherical module fitting inside a cell of the grid. At any point, each node occupies a cell, with the positioning of the nodes defining a shape, and two nodes may not occupy the same cell. It also defines an undirected *neighbouring relation* $E \subset V \times V$, where $uv \in E$ iff $o_x(u) = o_x(v)$ and $|o_y(u) - o_y(v)| = 1$ or $o_y(u) = o_y(v)$ and $|o_x(u) - o_x(v)| = 1$, that is, if u and v occupy *horizontally* or *vertically* adjacent cells of the grid. We use $N(u)$ to denote the set of neighbours of a node u in a given shape. A shape is *connected* if the graph induced by its neighbouring relation is a connected graph.

In general, shapes can *transform* to other shapes via a sequence of one or more mechanical operations, which we refer to as movements. We consider only one type of movement: *rotation*. In this movement, a single node moves relative to one or more neighbouring nodes. A single rotation movement of a node u is a 90° rotation of u around one of its neighbours. Let (x, y) be the current position of u and let its neighbour be v occupying the cell $(x, y-1)$. Then, u can rotate 90° clockwise (counterclockwise) around v iff the cells $(x+1, y)$ and $(x+1, y-1)$ ($(x-1, y)$ and $(x-1, y-1)$, respectively) are both empty. By rotating the whole system 90°, 180°, and 270°, all possible rotation movements can be defined. See Figure 1 for an example.

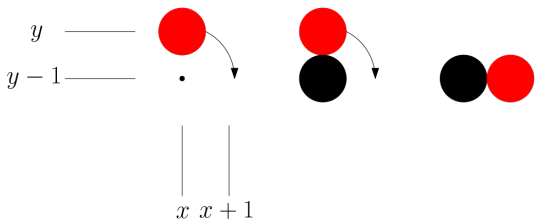

Figure 1 An example of rotation movement. A node on the black dot (in the row $y-1$) and empty cells at positions $(x+1, y)$ and $(x+1, y-1)$ are required for this movement.

Let A and B be two connected shapes. We say that A transforms to B via a rotation r, denoted $A \xrightarrow{r} B$, if there is a node u in A such that if u applies r, then the shape resulting after the rotation is B. We say that A transforms in one step to B (or that B is reachable in

one step from A), denoted $A \to B$, if $A \xrightarrow{r} B$ for some rotation r. We say that A transforms to B (or that B is reachable from A) if there is a sequence of shapes $A = S_1, S_2, \ldots, S_t = B$, such that $S_i \to S_{i+1}$ for all $1 \leq i \leq t-1$. Rotation is a reversible movement, a fact that we use in our results. As a condition of the problem we consider, all shapes $S_1, S_2, \ldots, S_t$ must be connected shapes.

At the start of each transformation, we will be assuming the existence of a *seed*: a small connected shape M placed on the perimeter of the given shape S to trigger the transformation. This is essential because under the constraints of a model with rotation-only movement, there are shapes S that are κ-*blocked*, meaning that at most κ movements can be made before a configuration is repeated.

For the sake of providing clarity to our transformations, we say that every cell in the two-dimensional grid has a colour from $\{red, black\}$ in such a way that the cells form a black and red checkered colouring of the grid, similar to the colouring of a chessboard. This represents a property of the rotation movement, which is that any given node in a coloured cell can only enter cells of the same colour. We define $c(u) \in \{black, red\}$ as the colour of the cell occupied by the node u for a given chessboard colouring of the grid. We represent this in our figures by colouring the nodes red or black.

Any shape S consists of $b(S)$ black and $r(S)$ red nodes. Two shapes A and B are *colour-consistent* if $b(A) = b(B)$ and $r(A) = r(B)$. For any shape S of n nodes, the *parity* of S is the colour of the majority of nodes in S. If there is no strict majority, we pick any as the parity colour.

We use σ and variants to denote sequences of nodes. A *k-subsequence* σ' of a sequence σ is any subsequence of σ where $|\sigma'| = k$. For a given colouring of the grid, the *colour sequence* $c(\sigma)$ of a sequence of nodes $\sigma = (u_1, u_2, \ldots, u_n)$ is defined as $c(\sigma) = (c(u_1), c(u_2), \ldots, c(u_n))$. A sequence σ' is *colour-order preserving with respect to* σ if $c(\sigma') = c(\sigma)$.

The *perimeter* of a connected shape S is the minimum-area polygon that completely encloses S in its interior, existence of an interior and exterior directly following from the Jordan curve theorem [18]. The *cell perimeter* of S consists of every cell of the grid not occupied by S that contributes at least one of its edges to the perimeter of S. The *external surface* of S consists of all nodes $u \in S$ such that u occupies a cell defining at least one of the edges of the perimeter of S. The *extended external surface* of S is defined by adding to the external surface all nodes of S whose cell shares a corner with the perimeter of S.

The orthogonal convex hull $H(S)$ of a connected shape S is defined as the intersection of all orthogonally convex shapes of which S is a subshape. Given a connected shape S, a *pocket* is a maximal connected set of empty cells exterior to the shape and interior to its orthogonal convex hull $H(S)$. The *boundary* of a pocket consists of a line segment which is a subchain of the perimeter of $H(S)$, called the *pocket lid*, and a subchain of the perimeter of the shape, called the *pocket subchain*.

Place a $c \times c$ square K on a subchain of the perimeter of S which is exterior to its convex hull and shift it around. A *$c \times c$-narrow pocket* is a maximal subset of a pocket of S which can never overlap with K. The *bottleneck* of a $c \times c$-narrow pocket is the subchain which separates the cells reachable by K from the cells within the pocket which are unreachable. The *$c \times c$-reachable boundary* of S is defined as the areas of the perimeter that are reachable by K, plus the bottlenecks. A shape S has a *$c \times c$-wide exterior* if it has no $c \times c$-narrow pockets. Throughout the paper, a *bottleneck lid* is the lid of a 4×4-narrow pocket.

A black parity (similarly for red parity) *line-with-leaves* [21] is a straight line with one or more black leaves attached to its red nodes. A *double-line-with-leaves* L is a shape obtained by joining together the endpoints of two lines-with-leaves L_1 and L_2 which have opposite

colour parities and the same orientation. Additionally, both endpoints of the red parity line-with-leaves must be black and both endpoints of the black parity line-with-leaves must be red. We call the straight line formed by joining the straight lines of L_1 and L_2 the *core line* of L.

5 The Universal Transformation

In this section, we give the technical details of the universal transformation. Given any two connected colour-consistent shapes S and S', our goal is to transform S into S'. We assume a seed M of $d = O(1)$ nodes, meaning that we are free to place any connected shape of d nodes on the perimeter of the shape. The transformation will establish the following theorem.

▶ **Theorem 26.** *Let S and S' be any two connected colour-consistent shapes. Then, there is a connected shape M of $d = O(1)$ nodes and a placement of M on the perimeter of S, such that $S \cup M$ can be transformed into S' via $O(n^2)$ rotation movements.*

To prove Theorem 26 it is sufficient to show that, for any shape S and some placement of a d-seed M on the perimeter of S, $S \cup M$ can be transformed into a double-line-with-leaves. By reversibility of rotation and the fact that we can transform any pair of colour-consistent double-lines-with-leaves into each other, it follows that any S can be transformed into any S' via the double-line-with-leaves canonical shape.

The following is an intuitive description of our strategy. We will show that there is a placement of the d-seed on the perimeter of S from which a "good" starting configuration for the transformation can be obtained. A 6-robot formed by the d-seed sets up the shape to be in a configuration having the following structures on the perimeter of the shape: (i) a "ladder" on which the double-line-with-leaves will be built, (ii) a reservoir of 7 nodes to be used for the compression subroutine, and (iii) a 6-robot. Then, as long as there are reachable nodes to be removed, the 6-robot picks a removable node, transports it (as a 7-robot), and places it on the double-line-with-leaves. This involves proving that the 6-robot (7-robot) can traverse the 4×4-reachable boundary (5×5, respectively) of S, by traversing reachable parts and bypassing unreachable parts, and that there is an order of removing nodes from the perimeter of S, until S is eliminated. This order, called an elimination sequence, removes small clusters of nodes such that no removal of a node disconnects the shape and nodes are only removed from the perimeter of the shape. Whenever there is no cluster of nodes that the 6-robot can reach, we show how to reconfigure S into another shape that has a reachable cluster of nodes. In particular, if no cluster of nodes exists on the perimeter that is also reachable, then there exists a pocket lid that we can close with auxiliary nodes, such that a cycle C on the perimeter is created. We can then compress C in a way that a cluster of nodes becomes reachable and, after removing the cluster, there exists a cycle C' on the perimeter of the shape. The 6-robot transports, one by one, the nodes of the cluster to the double-line-with-leaves, and removes any auxiliary nodes used to close a lid, before moving on to the next cluster.

5.1 Perimeter Traversal

We begin by showing that the 6-robot and the 7-robot can both traverse the perimeter of the shape, by visiting reachable parts and bypassing unreachable parts.

In [5], it was shown that a 6-robot and a 7-robot can both traverse the perimeter of an orthogonally convex shape. We say that a shape S, which is not orthogonally convex, is *traversable by orthogonally convex movement*, if the movements from [5] can be used to traverse its perimeter.

▶ **Theorem 1** ([5]). *For any orthogonally convex connected shape S, a 6-robot and a 7-robot are both capable of traversing the perimeter of S.*

As in [5], a 6-robot is a 3×2 group of connected nodes used to transport nodes around the shape. By using rotation movements, the 6-robot can slide across and climb lines of the perimeter. These movements of the robot are the outcome of a sequence of rotations, and should not be confused with the sliding of individual nodes of [14, 13, 20].

We prove that a 6-robot can traverse the 4×4-reachable boundary and a 7-robot can traverse the 5×5-reachable boundary of any connected shape S by orthogonally convex movement. Beginning with the 6-robot, our strategy is to show that shapes with 4×4-wide exteriors are traversable by orthogonally convex movement. We then show that for shapes with 4×4-narrow pockets, the robot can avoid entering those pockets by crossing them.

We assume that a 6-robot, which we will try to move around the perimeter of S, is given on the 4×4-reachable boundary as a 3×2 rectangle. For the 7-robot, we consider the 5×5-reachable boundary as the 7-robot requires more space to move. We first prove that shapes with a 4×4-wide exterior (5×5) have sufficient space for the 6-robot (7-robot, respectively) to perform the climbing and sliding movements of [5], through which the robot can traverse the perimeter of the shape.

▶ **Lemma 2.** *The perimeter of a connected shape S with a 4×4-wide exterior (5×5-wide exterior) can be traversed by orthogonally convex movement by a 6-robot (7-robot, respectively).*

Next, we show that it is possible to traverse small pockets by crossing them. We use the term *gap* to refer to the part of the pocket as well as lines of nodes neighbouring it, which are relevant for crossing operations. We first give our representation of the cases which we consider for these operations, as well as the variables we will be using to describe them.

Assume without loss of generality that the movement of the robot when crossing the gap is to the right, and if necessary, upwards. We have five coordinates: x_l, x_r, y_d, y_u and y_m (see Figure 2 in the Appendix). These coordinates in turn are used to calculate the three variables we use: $size$ is equal to $|x_r - x_l|$, $depth = |y_m - y_d|$ and $incline = |y_u - y_m|$. Intuitively, $size$ represents the horizontal distance between the first and last nodes of the gap, $depth$ represents the vertical distance between the first node of the gap and the bottom of the gap, and $incline$ represents the vertical distance between the first and last nodes of the gap. We say that a gap is *level* if $incline = 0$.

We first show that gaps of size ≤ 3 can be crossed by the 6-robot, by considering a set of cases which cover every possible situation. We assume the robot is above the gap in both the initial and final locations. This is to ensure that the movements do not need to make assumptions about the structure of the rest of the shape. We then prove that the robot can always reach this desired starting location, even in edge cases. It follows from this and from Theorem 1 that it is possible for the 6-robot to traverse the 4×4-reachable boundary of any shape. See Figure 3 in the Appendix for the main cases we consider in our proof, and Figures 4 and 5 for two examples of crossing a gap.

▶ **Lemma 3.** *The 6-robot can cross any gap of size ≤ 3 and the 7-robot any gap of size ≤ 4.*

▶ **Lemma 4.** *The 6-robot and the 7-robot can position themselves at the start of any gap.*

▶ **Theorem 5.** *The 4×4-reachable boundary (5×5-reachable boundary) of any connected shape S can be traversed by the 6-robot (7-robot, respectively).*

5.2 Elimination Sequence

In this section, we present the sequence in which the 6-robot transports the nodes of the shape S. The elimination sequence of [5] for orthogonally convex shapes, was designed to preserve connectivity and respect the colour capacity of the line-with-leaves, which was the canonical shape. For general connected shapes, we must additionally ensure that removed nodes should lie in a position that the 6-robot can reach. We first give an elimination sequence for the relaxed case in which nodes can be removed from anywhere on the extended external surface of the shape. We will then add the requirement that removed nodes must be reachable by a 6-robot. Apart from simplifying exposition, the relaxed elimination sequence could be useful to any future transformations that would somehow circumvent the reachability issue.

Let S be a connected shape. An *elimination sequence* $\sigma = (u_1, u_2, \ldots, u_n)$ of a shape S is a permutation of the nodes of S satisfying the following properties. Let $S_t = S_{t-1} \setminus \{u_t\}$, where $1 \leq t \leq n$ and $S_0 = S$. Observe that S_n is always the empty shape. The first property is that, for all $1 \leq t \leq n-1$, S_t must be a connected shape. Moreover, for all $1 \leq t \leq n$, u_t must be a node on the extended external surface of S_{t-1}. Essentially, σ defines a sequence $S = S_0[u_1]S_1[u_2]S_2[u_3]\ldots S_{n-1}[u_n]S_n = \emptyset$, where, for all $1 \leq t \leq n$, the connected shape S_t is obtained by removing node u_t from the extended external surface of shape S_{t-1}.

Our strategy for the elimination sequence is to remove nodes in small *clusters* that have some "nice" properties. Each such cluster will contains 2 to 4 nodes, and is either a pair of neighbouring nodes, or a node together with a subset of its neighbours, which must be leaves. We want to show that removing any of these clusters does not disconnect the shape, and that the clusters have specific colour properties. We first give some necessary definitions.

▶ **Definition 6.** *Let $S = (V, E)$ be a connected shape. Node $u \in V$ is a* separator *node if $S' = (V \setminus \{u\}, E')$, where $E' = E \setminus \{uv \in E \mid v \in N(u)\}$, is a disconnected shape.*

▶ **Definition 7.** *Let $S = (V, E)$ be a connected shape. Node u is a* local *separator node, if u is a separator node and there exists a subset $N'(u)$ of $N(u)$, such that $S' = (V', E')$, where $V' = V \setminus N'(u)$ and $E' = E \setminus \{uv \in E \mid v \in N'(u)\}$, is a connected shape and u is not a separator node in S'.*

▶ **Definition 8.** *We define a* cluster *of nodes C, to be a set of nodes on the extended external surface of a shape S that satisfies one of the following two properties:*

1. *C contains two neighbouring nodes u, v such that removing u and then v, or v and then u does not disconnect the shape.*
2. *C contains a local separator node u, and every neighbour of u that is a leaf in S.*

Intuitively, our algorithm computes an elimination sequence as follows. Operating in phases until the whole shape is eliminated, it marks the extended external surface of the shape and repeatedly finds and removes clusters of marked nodes. The order in which the nodes of a cluster are added to the elimination sequence is the order in which the nodes will be transported by the 6-robot. When there is no cluster of marked nodes, the algorithm moves on to the next phase, marking the new extended external surface and searching for clusters of marked nodes. The pseudocode is given in Algorithm 1 in the Appendix. Connectivity-preservation is guaranteed by the separator properties of the clusters.

▶ **Lemma 9.** *Let S be a connected shape on which we execute Algorithm 1 and let σ be the elimination sequence produced by the algorithm. For all $1 \leq t \leq n-1$, S_t is a connected shape.*

We also need to show that the algorithm terminates. We do this by making use of the fact that the extended external surface of a connected shape defines a cactus graph.

▶ **Definition 10.** *Given a connected shape $S = (V, E)$, we define a graph $S' = (V', E')$ on its external extended surface as follows. V' contains every node of the extended external surface and $uv \in E'$ iff u and v are consecutively visited in a clockwise walk on the perimeter of S.*

▶ **Lemma 11.** *The graph of Definition 10 is a cactus graph.*

▶ **Lemma 12.** *Let S be a connected shape on which we execute Algorithm 1. Algorithm 1 terminates and outputs a sequence $\sigma = (u_1, u_2, \ldots, u_n)$ that is a permutation of the nodes of S, where for all $1 \leq t \leq n$, u_t is a node on the extended external surface of S_{t-1}.*

▶ **Theorem 13.** *When executed on any connected shape S, Algorithm 1 terminates giving as output an elimination sequence of S.*

Proof. Follows from Lemmas 9 and 12. ◀

The following lemma will be later used to show that the order of colours of the elimination sequence produced by Algorithm 1 respects the colour capacity of a double-line-with-leaves.

▶ **Lemma 14.** *Consider a connected shape S on which we execute Algorithm 1 and let σ be the elimination sequence produced by the algorithm. There exists a way to split σ into consecutive subsequences $\sigma_1\sigma_2\cdots\sigma_k$ such that every subsequence contains consecutive nodes from σ and has one of the following colour sequences: bbbr,bbr,br,rrrb,rrb,rb.*

5.3 Adding Reachability

In the previous section, we showed that, in principle, there exists an elimination sequence. However, in the actual transformation the 6-robot must be able to reach the nodes to be removed. In this section, we show how to restructure a shape that has no cluster of nodes on the reachable part of the extended external surface, so that such a cluster becomes available.

▶ **Observation 15.** *There exist shapes such that no cluster of nodes on their extended external surface is reachable by the 6-robot. For example, trees concealing their endpoints within narrow spirals.*

In contrast to what holds for orthogonally convex shapes [5], we cannot hope to eliminate a general shape only by directly removing nodes from its extended external surface. As a consequence, further reconfiguration is needed and the elimination sequence of Algorithm 1 must be modified accordingly. First, we extend the definition of the extended external surface to account for nodes that are reachable by the 6-robot.

We say a node u in shape S is reachable if node u resides on the $c \times c$-reachable boundary of S. The *reachable external surface* of a connected shape A is a shape B, not necessarily connected, consisting of all nodes $u \in A$ such that u occupies a cell defining at least one of the line segments of A's $c \times c$-reachable boundary. The *reachable extended external surface* of a connected shape A, is defined by adding to A's reachable external surface all nodes of A whose cells share a corner with A's reachable boundary.

Whenever we have a shape S, where every cluster of nodes is not reachable by the 6-robot, we employ a restructuring strategy, where the 6-robot moves nodes around on shape S, until a cluster of nodes can be reached by the 6-robot. First, we modify Algorithm 1 so that the algorithm marks every node of the reachable extended external surface (see Algorithm 2 in the Appendix). This guarantees that every node of the elimination sequence is reachable. Additionally, whenever there exists no cluster of nodes on the reachable extended external

surface, we call a restructuring subroutine. After restructuring is finished, we mark every node of the reachable extended external surface and continue removing clusters. The above two steps are repeated until the shape is empty.

Our strategy for restructuring is based on compression (see Algorithm 3 in the Appendix). This involves the process of creating a cycle on the reachable extended external surface of the shape by adding some auxiliary nodes. Once this is achieved, we show how we can move some nodes on the extended reachable external surface of the shape such that (i) we "compress" the cycle on the extended external surface by removing some nodes from the cycle and making the cycle smaller and (ii) the removed nodes form a cluster of nodes that will reside on the reachable extended external surface of the shape. We show that we can always add auxiliary nodes to the shape, such that we create a cycle C on the extended external surface, where one of the *concave corner* nodes of C is reachable, and either is not a separator node or it is a local separator node.

▶ **Definition 16.** *Let C be a cycle on the extended external surface of a connected shape S. We say that a node u is a concave corner of cycle C if $u \in C$, u has two neighbouring nodes v, w, where $v, w \in C$, and the cell adjacent to both v and w is part of the interior of the shape.*

▶ **Definition 17.** *Consider a connected shape S that contains a pocket P.* Closing a lid L of pocket P *is the process of placing nodes in the empty cells of the pocket P that are adjacent to the pocket lid L.*

▶ **Lemma 18.** *The number of nodes needed to close a bottleneck lid is at most 7.*

▶ **Lemma 19.** *Consider any connected shape $S = (V, E)$, where no cluster of nodes resides on the reachable extended external surface. Then, there exists one pocket lid that can be closed such that the new shape S' has a cycle C on the extended external surface, where one of the concave corner nodes of C is both reachable and is either a local separator node or it is not a separator node.*

Using Lemma 19, we can show that after closing a lid, a cluster of nodes can be removed. However, since the number of auxiliary nodes needed to close a lid can be larger than the size of a cluster, this strategy is not guaranteed to succeed. To circumvent this, we use a compression technique starting from concave corner of the shape. After compressing the shape, we still have a cycle C on the extended external surface, where one of the concave corner nodes of C is both reachable and is either a local separator node or it is not a separator node. Additionally, after the compression, we will have a cluster of nodes on the reachable extended external surface that is not part of C. This allows us to compress at least once using the same auxiliary nodes, and then we can remove the auxiliary nodes that were used to close the lid.

▶ **Lemma 20.** *Consider any connected shape $S = (V, E)$ that has no cluster of nodes on its reachable extended external surface, where we close a lid with the auxiliary node set V_A that creates a cycle C on the extended external surface, where one of the corner nodes u_1 of C is reachable and is a local separator node. There exists a way to* compress *the shape such that the extended external surface contains a cycle C', where every auxiliary node is in C'. Additionally, the reachable extended external surface contains a cluster of nodes that is not part of C'.*

▶ **Theorem 21.** *Let S be a connected shape where no cluster of nodes on the extended external surface is reachable by the 6-robot. Executing Algorithm 3 on S reconfigures S into a connected shape S' that has a reachable cluster of nodes on the extended external surface.*

Proof. Algorithm 3 adds auxiliary nodes to a pocket lid in order to create a cycle C on the extended external surface of S. Lemma 19 guarantees that we can find pocket lid to close with auxiliary nodes, that also creates a reachable concave corner node of C. Then, Lemma 20 guarantees that we can use this node in order to compress the cycle C into a cycle C', such that a cluster of nodes is reachable by the 6-robot, and that cluster of nodes does not contain any node in C'. Since C' is a cycle, we can remove the auxiliary nodes without disconnecting the shape. ◀

5.4 Generation Sequence

Given a connected shape S of n nodes, a *generation sequence* $\sigma = (u_1, u_2, \ldots, u_n)$ of shape S is a permutation of the nodes of S satisfying the following properties. Let $S_t = S_{t-1} \cup \{u_t\}$, where $1 \leq t \leq n$ and $S_0 = \emptyset$. Observe that $S_n = S$. Any shape generation sequence also satisfies the following properties, which it shares with the shape elimination sequence. The first property is that, for all $1 \leq t \leq n-1$, S_t must be a connected shape. Moreover, for all $1 \leq t \leq n$, u_t must be placed in the cell perimeter of S_{t-1}. Essentially, σ defines a sequence $\emptyset = S_0[u_1]S_1[u_2]S_2[u_3]\ldots S_{n-1}[u_n]S_n = S$, where, for all $1 \leq t \leq n$, a connected shape S_t is obtained by adding the node u_t to the cell perimeter of S_{t-1}. The generation sequence that we are going to compute, constructs a double-line-with-leaves.

Given as input an elimination sequence by Algorithm 2, Algorithm 4 will return a generation sequence that constructs a double-line-with-leaves. The algorithm, first constructs the unique bi-coloured pair of the core line and then extends it by placing nodes on both sides of the line.

The algorithm constructs a straight double-line-with-leaves, expects the first two nodes to be a bi-coloured pair and every subsequence to arrive afterwards to have one of the colour sequences as described in Lemma 14. The algorithm positions the first bi-coloured pair horizontally with the red coloured node on the left and the black coloured node on the right. Let (x_l, y_0) be the position of the leftmost node on the core line of the double-line-with-leaves and (x_r, y_0) be the position of the rightmost node on the core line of the double-line-with-leaves.

Every subsequence has size 2, 3 or 4 and has colours $br, bbr, bbbr$ or $rb, rrb, rrrb$. If a subsequence arrives starting with a black node (possible subsequences are $br, bbr, bbbr$), the first black node is placed at position (x_{l-1}, y_0), the last node (which must be red), is placed at (x_{l-2}, y_0), and any possible other black nodes are placed at positions $(x_l, y_1), (x_l, y_{-1})$. If a subsequence arrives starting with a red node (possible subsequences are rb, rrb, $rrrb$), the first red node is placed at position (x_{r+1}, y_0), the last node (which must be black), is placed at (x_{r+2}, y_0), and any possible other red nodes are placed at positions $(x_r, y_1), (x_r, y_{-1})$. The algorithm preserves the invariant that after the placement of the first bi-coloured pair and after the placement of every subsequent subsequence that arrives, the leftmost and rightmost positions of the line, called (x_l, y_0) and (x_r, y_0), have red and black nodes, respectively, and positions $(x_r, y_1), (x_r, y_{-1}), (x_r, y_1), (x_r, y_{-1})$ are empty. See Algorithm 4 in the Appendix for the pseudocode.

▶ **Lemma 22.** *Let σ be a bicoloured sequence of nodes that fulfils all the following conditions:*

- *The set of the first two nodes in σ is bi-coloured.*
- *σ can be split into consecutive subsequences $\sigma_1\sigma_2\cdots\sigma_k$ such that every subsequence contains consecutive nodes from σ and also has one of the following colour sequences: $bbbr, bbr, br, rrrb, rrb, rb$.*

Then there is a double-line-with-leaves generation sequence $\sigma' = (u'_1, u'_2, \ldots, u'_n)$ which is colour-order preserving with respect to σ.

▶ **Theorem 23.** *Given a shape elimination sequence σ computed by Algorithm 2, Algorithm 4 returns a generation sequence which is colour-order-preserving with respect to σ and which constructs a double-line-with-leaves.*

Proof. Follows from Lemmas 14 and 22. ◀

5.5 Wrapping up

To complete the result, we show how the transformation is initialised, including where the double-line-with-leaves will be constructed, and argue that picking the next node in the elimination sequence and placing it on the double-line-with-leaves is always possible.

At the beginning of the transformation, the 6-robot transports 5 nodes from the original seed, and places them as a straight vertical path of length 5, called a *ladder* atop one of the topmost nodes of the initial shape. Then, the first bi-coloured pair that arrives to the elimination sequence is placed perpendicular to the ladder and the double-line-with-leaves extends perpendicular to the ladder. This ladder of 5 nodes guarantees that the construction of the double-line-with-leaves will never create any narrow pockets and this implies that the 6-robot and 7-robot can reach any part of the double-line-with-leaves.

We now show that it is possible to remove nodes from a shape S in the order of a shape elimination sequence generated by Algorithm 2, and place them on the double-line-with-leaves L in the order of a double-line-with-leaves generation sequence created by Algorithm 4, crossing the ladder in the process.

▶ **Lemma 24.** *Given a shape elimination sequence σ generated by Algorithm 2 for a shape S which starts with the node u, and a double-line-with-leaves generation sequence σ' generated by Algorithm 4 for a double-line-with-leaves L, the 6-robot is able to traverse the 4×4 reachable boundary of $S \cup L$, pick u up, and become a 7-robot. It can then traverse the 5×5 reachable boundary of $(S \setminus \{u\}) \cup L$ and place u on L.*

▶ **Theorem 25.** *Let σ be the shape elimination sequence generated by Algorithm 2 for a shape S, and a double-line-with-leaves generation sequence σ' which is colour-order preserving with respect to σ (as generated by Algorithm 4), the 6-robot can remove nodes from S according to σ and construct the double-line-with-leaves according to σ'.*

Putting everything together, including the fact that a 6-robot can be used to transform any pair of colour-consistent double-lines-with-leaves into each other, we get:

▶ **Theorem 26.** *Let S and S' be any connected colour-consistent shapes. Then, there is a connected shape M of $d = O(1)$ nodes and a placement of M on the perimeter of S, such that $S \cup M$ can be transformed into S' via $O(n^2)$ rotation movements.*

6 Conclusions

We have shown that by using a seed of constant size, it is possible to transform any pair of connected shapes A and B on a two-dimensional square grid into each other in an optimal $O(n^2)$ time. This leaves a few open problems to be addressed. First, the issue of creating a distributed version of the algorithm. This will not only make the algorithm more immediately applicable to real-world programmable matter scenarios, which usually assume that each module acts independently, but also opens up the possibility of a "pipelined" or parallel version which may be able to perform the transformation in only $O(n)$ parallel time movements.

Another issue is the size of the seed. It may be possible to reduce the seed's size to as little as 3 nodes, which would be equivalent to [5]. However, if all removable nodes are concealed then this might not be enough, unless a new approach, possibly by "drilling" into boundaries of the shape, is adopted.

Third, a potential comparison could be made to the pivoting model result of [1], to compare the similarities and differences of each approach to universal transformation. Finally, it may be possible to extend the results from shapes on a two-dimensional square grid, to those in a three-dimensional environment. Universal transformation in a three-dimensional environment which does not disconnect the shape is another challenging goal with interesting potential applications.

References

1 Hugo A. Akitaya, Esther M. Arkin, Mirela Damian, Erik D. Demaine, Vida Dujmović, Robin Flatland, Matias Korman, Belen Palop, Irene Parada, André van Renssen, and Vera Sacristán. Universal Reconfiguration of Facet-Connected Modular Robots by Pivots: The O(1) Musketeers. *Algorithmica*, 83(5):1316–1351, May 2021. doi:10.1007/s00453-020-00784-6.

2 Nada Almalki and Othon Michail. On geometric shape construction via growth operations. *Theoretical Computer Science*, 984:114324, February 2024. doi:10.1016/j.tcs.2023.114324.

3 Abdullah Almethen, Othon Michail, and Igor Potapov. Pushing lines helps: Efficient Universal Centralised Transformations for Programmable Matter. *Theoretical Computer Science*, 830-831:43–59, August 2020. doi:10.1016/j.tcs.2020.04.026.

4 Dana Angluin, James Aspnes, Zoë Diamadi, Michael J. Fischer, and René Peralta. Computation in Networks of Passively Mobile Finite-State Sensors. *Distributed Computing*, 18(4):235–253, March 2006. doi:10.1007/s00446-005-0138-3.

5 Matthew Connor and Othon Michail. Centralised connectivity-preserving transformations by rotation: 3 musketeers for all orthogonal convex shapes. In *Proceedings of the 18th International Symposium on Algorithmics of Wireless Networks (ALGOSENSORS)*, pages 60–76, 2022.

6 Matthew Connor, Othon Michail, and Igor Potapov. Centralised Connectivity-Preserving Transformations for Programmable Matter: A Minimal Seed Approach. *Theoretical Computer Science*, November 2022. doi:10.1016/j.tcs.2022.09.016.

7 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(1):81–96, March 2018. doi:10.1007/s11047-017-9658-6.

8 Zahra Derakhshandeh, Shlomi Dolev, Robert Gmyr, Andréa W. Richa, Christian Scheideler, and Thim Strothmann. Amoebot - a new model for programmable matter. In *Proceedings of the 26th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 220–222, 2014. doi:10.1145/2612669.2612712.

9 Zahra Derakhshandeh, Robert Gmyr, Andrea W. Richa, Christian Scheideler, and Thim Strothmann. Universal Shape Formation for Programmable Matter. In *Proceedings of the 28th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 289–299, 2016. doi:10.1145/2935764.2935784.

10 Zahra Derakhshandeh, Robert Gmyr, Andréa W. Richa, Christian Scheideler, and Thim Strothmann. An Algorithmic Framework for Shape Formation Problems in Self-Organizing Particle Systems. In *Proceedings of the Second Annual International Conference on Nanoscale Computing and Communication (NANOCOM)*, pages 1–2, 2015. doi:10.1145/2800795.2800829.

11 David Doty. Theory of Algorithmic Self-Assembly. *Communications of the ACM*, 55(12):78–88, December 2012. doi:10.1145/2380656.2380675.

12 David Doty. Timing in Chemical Reaction Networks. In *Proceedings of the 2014 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, Proceedings, pages 772–784. December 2013. doi:10.1137/1.9781611973402.57.

13 Adrian Dumitrescu and János Pach. Pushing squares around. In *Proceedings of the Twentieth ACM Annual Symposium on Computational Geometry (SCG)*, pages 116–123, 2004.

14 Adrian Dumitrescu, Ichiro Suzuki, and Masafumi Yamashita. Motion planning for metamorphic systems: Feasibility, decidability, and distributed reconfiguration. *IEEE Transactions on Robotics and Automation*, 20(3):409–418, 2004.

15 Sándor P. Fekete, Phillip Keldenich, Ramin Kosfeld, Christian Rieck, and Christian Scheffer. Connected coordinated motion planning with bounded stretch. *Autonomous Agents and Multi-Agent Systems*, 37(2):43, October 2023.

16 Michael Feldmann, Andreas Padalkin, Christian Scheideler, and Shlomi Dolev. Coordinating Amoebots via Reconfigurable Circuits. *Journal of Computational Biology: A Journal of Computational Molecular Cell Biology*, 29(4):317–343, April 2022. doi:10.1089/cmb.2021.0363.

17 Kyle Gilpin, Ara Knaian, and Daniela Rus. Robot Pebbles: One Centimeter Modules for Programmable Matter through Self-Disassembly. In *2010 IEEE International Conference on Robotics and Automation*, pages 2485–2492, 2010. doi:10.1109/ROBOT.2010.5509817.

18 Camille Jordan. *Cours d'analyse de l'École polytechnique*, volume 1. Gauthier-Villars et fils, 1893.

19 George B. Mertzios, Othon Michail, George Skretas, Paul G. Spirakis, and Michail Theofilatos. The complexity of growing a graph. In *18th International Symposium on Algorithmics of Wireless Networks (ALGOSENSORS)*, pages 123–137, 2022.

20 Othon Michail, George Skretas, and Paul G. Spirakis. On the transformation capability of feasible mechanisms for programmable matter. In *44th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 136:1–136:15, 2017. doi:10.4230/LIPIcs.ICALP.2017.136.

21 Othon Michail, George Skretas, and Paul G. Spirakis. On The Transformation Capability of Feasible Mechanisms for Programmable Matter. *Journal of Computer and System Sciences*, 102:18–39, June 2019. doi:10.1016/j.jcss.2018.12.001.

22 Othon Michail and Paul G. Spirakis. Simple and Efficient Local Codes for Distributed Stable Network Construction. *Distributed Computing*, 29(3):207–237, June 2016. doi:10.1007/s00446-015-0257-4.

23 Matthew J. Patitz. An introduction to tile-based self-assembly and a survey of recent results. *Natural Computing*, 13:195–224, June 2014.

24 Paul W. K. Rothemund. Folding DNA to Create Nanoscale Shapes and Patterns. *Nature*, 440(7082):297–302, March 2006. doi:10.1038/nature04586.

25 Paul W. K. Rothemund and Erik Winfree. The Program-size Complexity of Self-Assembled Squares (extended abstract). In *Proceedings of the thirty-second Annual ACM Symposium on Theory of Computing (STOC)*, pages 459–468, May 2000. doi:10.1145/335305.335358.

26 Michael Rubenstein, Alejandro Cornejo, and Radhika Nagpal. Programmable Self-assembly in a Thousand-Robot Swarm. *Science*, 345(6198):795–799, August 2014. doi:10.1126/science.1254295.

27 David Soloveichik, Matthew Cook, Erik Winfree, and Jehoshua Bruck. Computation with Finite Stochastic Chemical Reaction Networks. *Natural Computing*, 7(4):615–633, December 2008. doi:10.1007/s11047-008-9067-y.

28 Pierre Thalamy, Benoit Piranda, and Julien Bourgeois. Distributed Self-Reconfiguration using a Deterministic Autonomous Scaffolding Structure. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 140–148, 2019.

29 Pierre Thalamy, Benoît Piranda, and Julien Bourgeois. 3D Coating Self-Assembly for Modular Robotic Scaffolds. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11688–11695, 2020. ISSN: 2153-0866. doi:10.1109/IROS45743.2020.9341324.

30 Erik Winfree. *Algorithmic Self-assembly of DNA*. PhD thesis, California Institute of Technology, June 1998.

31 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)*, pages 353–354, 2013. doi:10.1145/2422436.2422476.

A Appendix

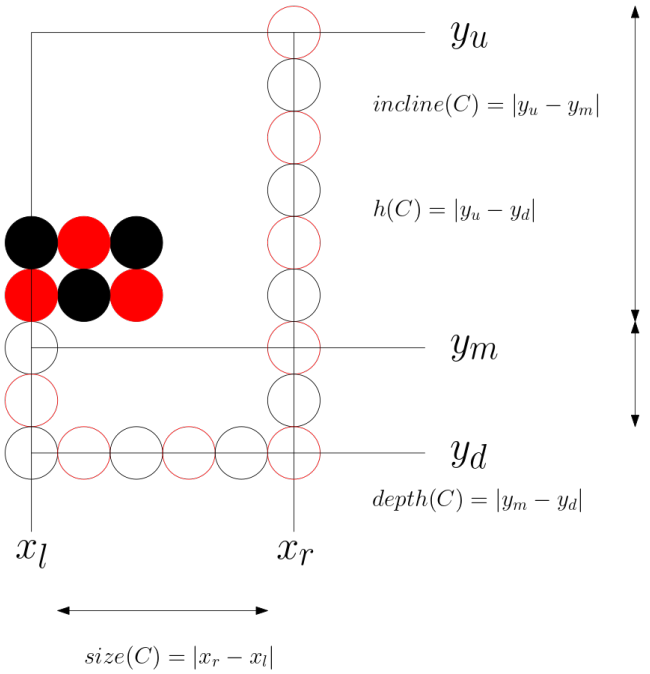

Figure 2 A visual representation of a gap, with the variables we use in our proof.

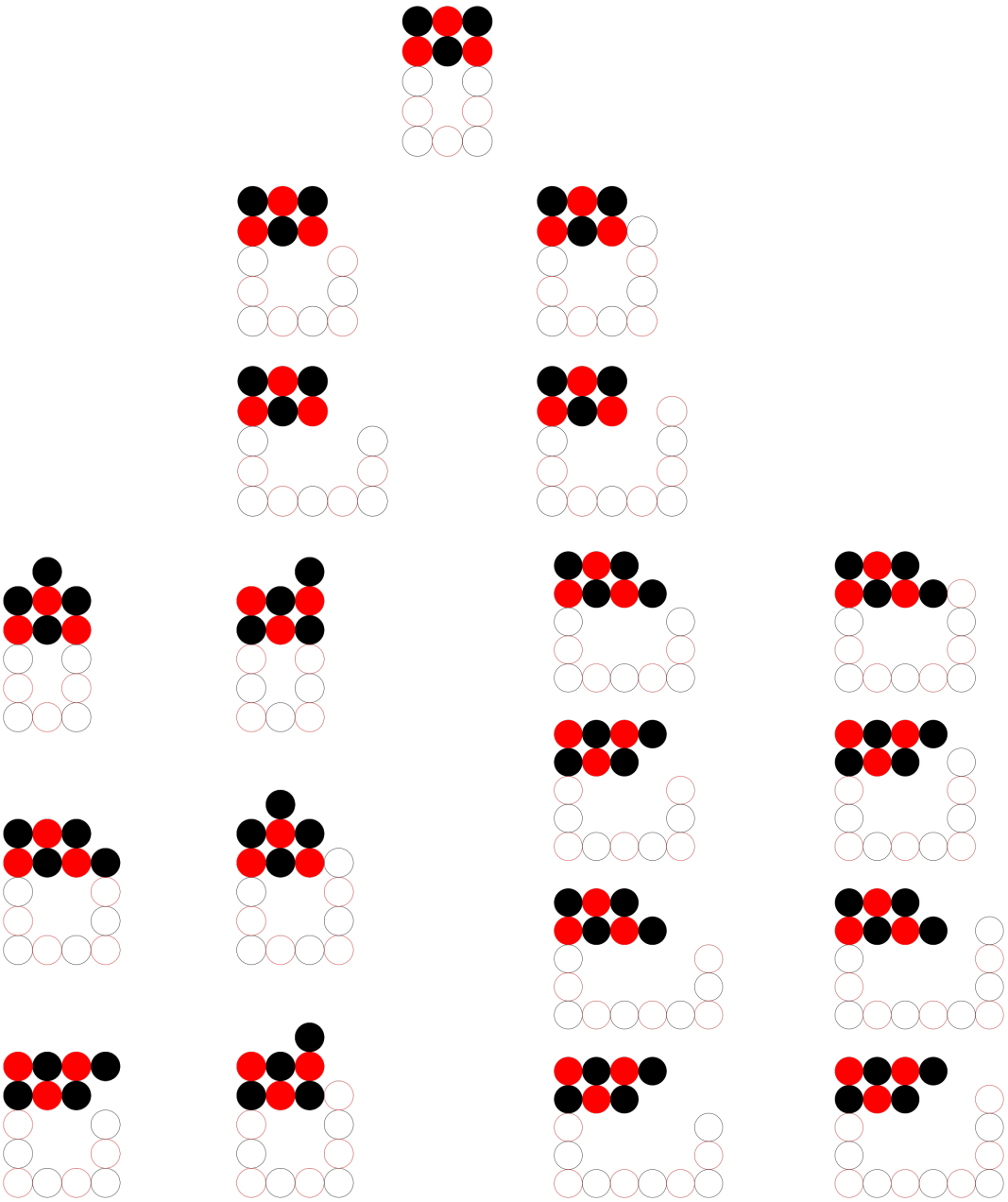

Figure 3 The main cases we consider in our proof. For all cases where the *incline* > 1, we only need to show that the robot can reach the other side of the gap, all movement afterwards is equivalent to climbing movements from [5]. All cases with *incline* < 1 are mirrored versions of cases with *incline* > 1. Our movements generally do not depend on connectivity with the bottom, so *depth* is mostly irrelevant, with the exception of a single edge case. All gaps with a size greater than those considered here are part of the 4×4 or 5×5-reachable boundary, for the 6-robot and 7-robot respectively, and per Lemma 2 can be traversed.

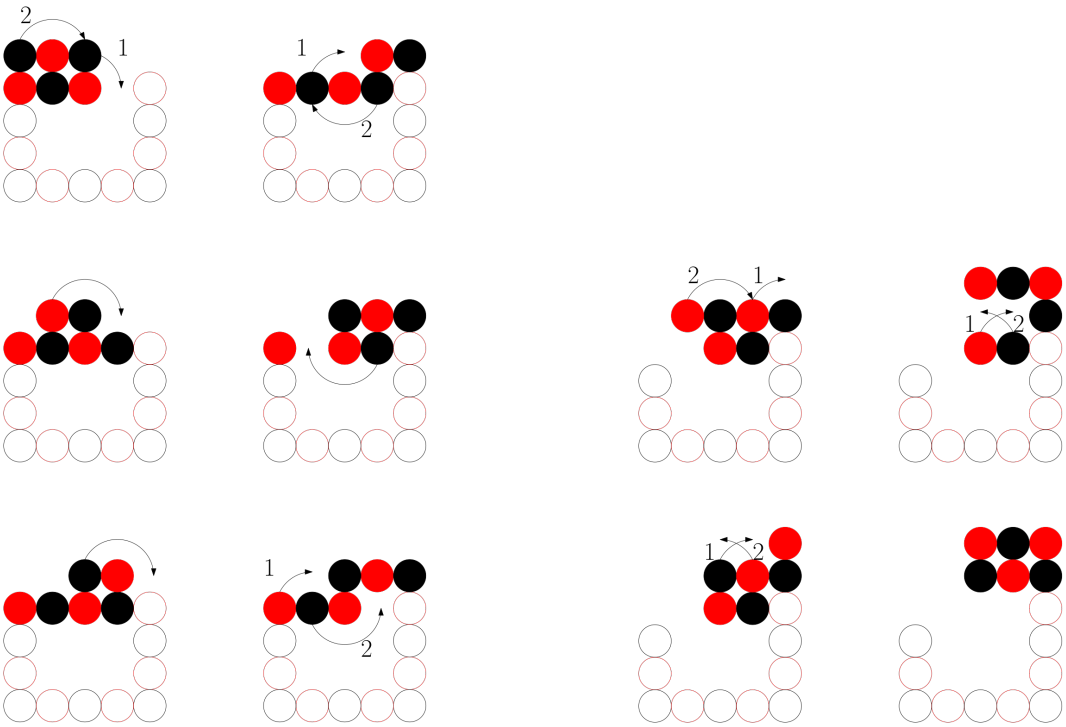

Figure 4 3-node inclined pocket traversal for $incline = 1$. Figures are read in columns, top-down.

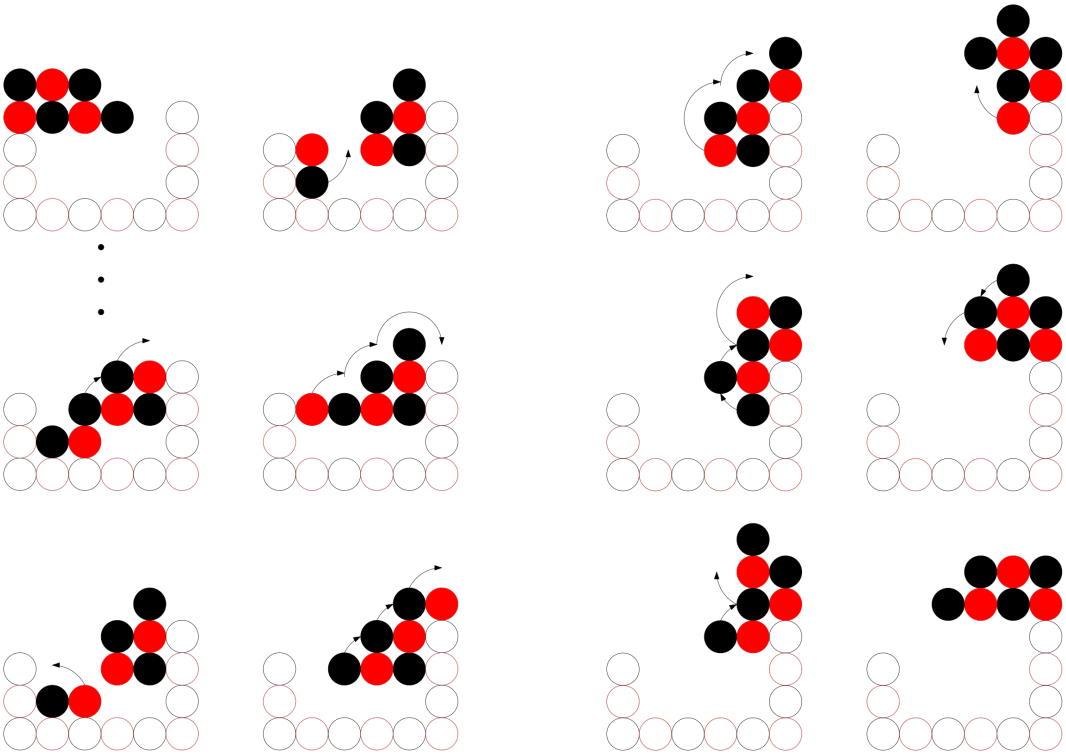

Figure 5 4-node pocket traversal for the 7-robot with the load in the bottom position with $incline = 1$.

Algorithm 1 Algorithm that computes a shape elimination sequence which does not require the nodes to be reachable by the 6-robot.

Input: Connected shape $S = (V, E)$
Output: Elimination sequence σ
while (S is not empty) **do**
  Mark every node u that belongs to the extended external surface of S;
  while (there exists a marked pair of nodes u, v satisfying property 1 of Definition 8) **do**
    Remove u, v from S and append them to σ in the order specified by Definition 8;
  end
  while (there exists a marked node u satisfying property 2 of Definition 8) **do**
    Remove the leaf neighbours of u from S and append them to σ;
    Remove u from S and append it to σ;
  end
  Unmark every marked node of S;
end
return σ

Algorithm 2 Algorithm that computes a shape elimination sequence.

Input: Connected shape $S = (V, E)$, $k = 0$
Output: Elimination sequence σ
while ($S \neq \emptyset$) **do**
  Mark every node that belongs to the reachable extended external surface of S;
  $k = 0$;
  while (there exists a marked pair of nodes u, v satisfying property 1 of Definition 8) **do**
    $k++$;
    Remove u, v from S and append them to σ in the order specified by Definition 8;
  end
  while (there exists a marked node $u \in S$ satisfying property 2 of Definition 8) **do**
    $k++$;
    Remove the leaf neighbours of u from S and append them to σ;
    Remove u from S and append it to σ;
  end
  Unmark every marked node $u \in S$;
  if $k = 0$ **then**
    Call Algorithm 3 with input $S = (V, E)$;
  end
end
return σ

Algorithm 3 Algorithm that reconfigures a connected shape to have a cluster of nodes on the reachable extended external surface.

Input: Connected shape $S = (V, E)$ with no cluster of nodes on the reachable extended external surface
Output: Connected shape $S' = (V, E')$ with a cluster of nodes on the reachable extended external surface

Close a lid using up to 7 auxiliary nodes to create a cycle C with the properties of Lemma 19;
Compress the cycle C;
Remove the auxiliary nodes used to close the lid;
return S'

Algorithm 4 Algorithm that constructs a double-line-with-leaves generation sequence.

Input: Elimination Sequence σ split into subsequences
Output: Double-line-with-leaves generation sequence $\sigma' = (u'_1, u'_2, \ldots, u'_n)$ which is colour-order preserving with respect to σ

if $c(u_1) == red$ *and* $c(u_2) == black$ **then**
  $u'_1 = (x_l, y_0)$, $u'_2 = (x_{l+1}, y_0)$;
else
  $u'_1 = (x_{l+1}, y_0)$, $u'_2 = (x_l, y_0)$;
end
Remove u_1, u_2 from σ;
$i = 3, b = 0, r = 0$;
while $\sigma \neq \emptyset$ **do**
  Remove the next subsequence σ_j from σ;
  if $c(u_i) == black$ **then**
    $u'_i = (x_{l-b-1}, y_0)$;
    if $|\sigma_j| == 2$ **then**
      $u'_{i+1} = (x_{l-b-2}, y_0)$;
    else if $|\sigma_j| == 3$ **then**
      $u'_{i+1} = (x_{l-b}, y_1)$, $u'_{i+2} = (x_{l-b-2}, y_0)$;
    else if $|\sigma_j| == 4$ **then**
      $u'_{i+1} = (x_{l-b}, y_1)$, $u'_{i+2} = (x_{l-b}, y_{-1})$, $u'_{i+3} = (x_{l-b-2}, y_0)$;
    $b = b + 2$, $i = i + |\sigma_j|$;
  else if $c(u_i) == red$ **then**
    - $u'_i = (x_{l+1+r+1}, y_0)$;
    if $|\sigma_j| == 2$ **then**
      $u'_{i+1} = (x_{l+1+r+2}, y_0)$;
    else if $|\sigma_j| == 3$ **then**
      $u'_{i+1} = (x_{l+1+r}, y_1)$, $u'_{i+2} = (x_{l+1+r+2}, y_0)$;
    else if $|\sigma_j| == 4$ **then**
      $u'_{i+1} = (x_{l+1+r}, y_1)$, $u'_{i+2} = (x_{l+1+r}, y_{-1})$, $u'_{i+3} = (x_{l+1+r+2}, y_0)$;
    $r = r + 2$, $i = i + |\sigma_j|$;
  end
end