On the Transformation Capability of Feasible Mechanisms for Programmable Matter[☆]

Othon Michail[a,*], George Skretas[a], Paul G. Spirakis[a,b,c]

[a] *Department of Computer Science, University of Liverpool, UK*
[b] *Computer Technology Institute & Press "Diophantus" (CTI), Patras, Greece*
[c] *Computer Engineering and Informatics Department (CEID), University of Patras, Greece*

Abstract

In this work, we study theoretical models of *programmable matter* systems. The systems under consideration consist of spherical modules, that are kept together by magnetic or electrostatic forces and are able to perform two minimal mechanical operations (or movements): *rotate* around a neighbor and *slide* over a line. In terms of modeling, there are n nodes arranged in a 2-dimensional grid that form some initial *shape*. The goal is for the initial shape A to *transform* to some target shape B by a sequence of movements. Most of the paper focuses on *transformability* questions, meaning whether it is in principle feasible to transform a given shape to another. We first consider the case in which only rotation is available to the nodes. Our main result is that deciding whether two given shapes A and B can be transformed to each other, is in **P**. We then insist on rotation only and impose the restriction that the nodes must maintain global connectivity throughout the transformation. We prove that the corresponding transformability question is in **PSPACE** and study the problem of determining the minimum *seeds* that can make feasible, otherwise infeasible transformations. Next we allow both rotations and slidings and prove universality: any two connected shapes A, B of the same number of nodes can be transformed to each other without breaking connectivity. The worst-case number of movements of the generic strategy is $\Theta(n^2)$. We improve this to $O(n)$ parallel time by a pipelining strategy, and prove optimality of both by matching lower bounds. We next turn our attention to distributed transformations. The nodes are now distributed processes able to perform communicate-compute-move rounds. We provide distributed algorithms for a general type of transformation.

Keywords:
programmable matter, transformation, reconfigurable robotics, shape formation, complexity, distributed algorithms

1. Introduction

Programmable matter refers to any type of matter that can *algorithmically* change its physical properties. "Algorithmically" means that the change (or *transformation*) is the result of executing an *underlying program*. Depending on the implementation, the program could either be a *centralized algorithm* capable of controlling the whole programmable matter system (*external* control) or a *decentralized protocol* stored in the material itself and executed by various sub-components of the system (*internal* control). For a concrete example, imagine a material formed by a collection of spherical (nano- or micro-) modules kept together by magnetic or electrostatic forces. Each module is capable of storing (in some internal representation) and executing a simple program that handles communication with nearby modules and that controls the module's

[☆] A preliminary version of the results in this paper has appeared in [MSS17a].

[*] Corresponding author (Telephone number: +44 (0)151 795 4275, Postal Address: Department of Computer Science, University of Liverpool, Ashton Street, Liverpool L69 3BX, UK).

Email addresses: Othon.Michail@liverpool.ac.uk (Othon Michail), G.Skretas@liverpool.ac.uk (George Skretas), P.Spirakis@liverpool.ac.uk (Paul G. Spirakis)

electromagnets/capacitors, in a way that allows the module to *rotate* or *slide* over neighboring modules. Such a material would be able to adjust its *shape* in a programmable way. Other examples of physical properties of interest for real applications would be connectivity, color [LYS+01, CLS+11], and strength of the material.

Computer scientists, nanoscientists, and engineers are more and more joining their forces towards the development of such programmable materials and have already produced some first impressive outcomes (even though it is evident that there is much more work to be done in the direction of real systems), such as programmed DNA molecules that self-assemble into desired structures [Rot06, Dot12] and large collectives of tiny identical robots that orchestrate resembling a single multi-robot organism (e.g., the Kilobot system [RCN14]). Other systems for programmable matter include the Robot Pebbles [GKR10], consisting of 1cm cubic programmable matter modules able to form 2-dimensional (abbreviated "2D" throughout) shapes through self-disassembly, and the Millimotein [KCL+12], a chain of programmable matter which can fold itself into digitized approximations of arbitrary 3-dimensional (abbreviated "3D" throughout) shapes. Ambitious long-term applications of programmable materials include molecular computers, collectives of nanorobots injected into the human circulatory system for monitoring and treating diseases, or even self-reproducing and self-healing machines.

Apart from the fact that systems work is still in its infancy, there is also an apparent lack of unifying formalism and theoretical treatment. Still there are some first theoretical efforts aiming at understanding the fundamental possibilities and limitations of this prospective. The area of *algorithmic self-assembly* tries to understand how to program molecules (mainly DNA strands) to manipulate themselves, grow into machines and at the same time control their own growth [Dot12]. The theoretical model guiding the study in algorithmic self-assembly is the Abstract Tile Assembly Model (aTAM) [Win98, RW00] and variations. Recently, a model, called the *nubot* model, was proposed for studying the complexity of self-assembled structures with active molecular components [WCG+13]. This model "is inspired by biology's fantastic ability to assemble biomolecules that form systems with complicated structure and dynamics, from molecular motors that walk on rigid tracks and proteins that dynamically alter the structure of the cell during mitosis, to embryonic development where large-scale complicated organisms efficiently grow from a single cell" [WCG+13]. Another very recent model, called the *Network Constructors* model, studied what stable networks can be constructed by a population of finite-automata that interact randomly like molecules in a well-mixed solution and can establish bonds with each other according to the rules of a common small protocol [MS16]. The development of Network Constructors was based on the *Population Protocol* model of Angluin *et al.* [AAD+06], that does not include the capability of creating bonds and focuses more on the computation of functions on inputs. A very interesting fact about population protocols is that, when operating under a uniform random scheduler, they are formally equivalent to a restricted version of stochastic *chemical reaction networks* (CRNs), "which model chemistry in a *well-mixed solution* and are widely used to describe information processing occurring in natural cellular regulatory networks" (see, e.g., [SCWB08, Dot14]). Also the recently proposed *Amoebot* model, "offers a versatile framework to model self-organizing particles and facilitates rigorous algorithmic research in the area of programmable matter" [DDG+14, DGR+15, DDG+18, DGR+16, LFS+17]. See also [FRRS16] for latest activities and developments in this area of research.

Each theoretical approach, and to be more precise, each individual model, has its own beauty and has lead to different insights and developments regarding potential programmable matter systems of the future and in some cases to very intriguing technical problems and open questions. Still, it seems that the right way for theory to boost the development of more refined real systems is to reveal the *transformation capabilities of mechanisms and technologies that are available now*, rather than by exploring the unlimited variety of theoretical models that are not expected to correspond to a real implementation in the near future.

In this paper, we follow such an approach, by studying the transformation capabilities of models for programmable matter, which are based on minimal mechanical capabilities, easily implementable by existing technology.

1.1. Our Approach

We study a minimal programmable matter system consisting of n cycle-shaped modules, with each module (or *node*) occupying at any given time a cell of the 2D grid (no two nodes can occupy the same cell at the

same time). Therefore, the composition of the programmable matter systems under consideration is discrete. Our main question throughout is whether an initial arrangement of the material can *transform* (either in principle, e.g., by an external authority, or by itself) to some other target arrangement. In more technical terms, we are provided with an *initial shape* A and a *target shape* B and we are asked whether A *can be transformed to* B via a sequence of *valid* transformation steps. Usually, a step consists either of a *valid movement* of a single node (in the *sequential case*) or of more than one node at the same time (in the *parallel case*). We consider two quite primitive types of movement. The first one, called *rotation*, allows a node to rotate 90° around one of its neighbors either clockwise or counterclockwise (see, e.g., Figure 1 in Section 2) and the second one, called *sliding*, allows a node to slide by one position "over" two neighboring nodes (see, e.g., Figure 2 in Section 2). Both movements succeed only if the whole direction of movement is free of obstacles (i.e., other nodes blocking the way). More formal definitions are provided in Section 2. One part of the paper focuses on the case in which only rotation is available to the nodes and the other part studies the case in which both rotation and sliding are available. The latter case has been studied to some extent in the past in the, so called, *metamorphic systems* [DSY04a, DSY04b, DP04], which makes those studies the closest to our approach.

For rotation only, we introduce the notion of *color-consistency* and prove that if two shapes are not color-consistent then they cannot be transformed to each other. On the other hand color-consistency does not guarantee transformability, as there is an infinite set of pairs (A, B) such that A and B are color consistent but still cannot be transformed to each other. At this point, observe that if A can be transformed to B then the inverse is also true, as all movements considered in this paper are *reversible*. We distinguish two main types of transformations: those that are allowed to break the connectivity of the shape during the transformation and those that are not; we call the corresponding problems Rot-Transformability and RotC-Transformability, respectively. We prove that RotC-Transformability is a proper subset of Rot-Transformability by showing that a line-folding problem is in Rot-Transformability\RotC-Transformability. Our main result regarding Rot-Transformability is that Rot-Transformability $\in \mathbf{P}$. To prove polynomial-time decidability, we prove that two shapes A and B are transformable to each other iff both A and B have at least one movement available (without any movement available, a shape is *blocked* and can only trivially transform to itself). Therefore, transformability reduces to checking the availability of a movement in the initial and target shapes. The idea is that if a movement is available in a shape A, then there is always a way to *extract* from A a *2-line* (meaning to disconnect a line of length 2 from shape A, via a sequence of rotation movements). Such a 2-line can move freely in any direction and can also extract further nodes to form a *4-line*. A 4-line in turn can also move freely in any direction and is also capable of extracting nodes from the shape and transferring them, one at a time, to any desired target position. In this manner, the 4-line can transform A into a line with leaves around it that is color-consistent to A (based on a proposition that we prove, stating that any shape has a corresponding color-consistent line-with-leaves). Similarly, B, given that it is color-consistent with A, can be transformed by the same approach to exactly the same line-with-leaves, and then, by reversibility, it follows that A and B can be transformed to each other by using the line-with-leaves as an intermediate. This set of transformations do not guarantee the preservation of connectivity during the transformation. That is, even though the initial and target shapes considered are connected shapes, the shapes formed at intermediate steps of the transformation may very well be disconnected shapes.

We next study RotC-Transformability, in which again the only available movement is rotation, but now connectivity of the material has to be preserved throughout the transformation. The property of preserving the connectivity is expected to be a crucial property for programmable matter systems, as it allows the material to maintain coherence and strength, to eliminate the need for wireless communication, and, finally, enables the development of more effective power supply schemes, in which the modules can share resources or in which the modules have no batteries but are instead constantly supplied with energy by a centralized source (or by a supernode that is part of the material itself). Such benefits can lead to simplified designs and potentially to reduced size of individual modules. We first prove that RotC-Transformability $\in$ **PSPACE**. The rest of our results here are strongly based on the notion of a *seed*. This stems from the observation that a large set of infeasible transformations become feasible by introducing

to the initial shape an additional, and usually quite small, seed; i.e., a small shape that is being attached to some point of the initial shape in order to trigger an otherwise infeasible transformation. In particular, we prove that a *3-line seed*, if placed appropriately, is sufficient to achieve folding of a line (otherwise impossible). We then investigate seeds that could serve as components capable of traveling the perimeter of an arbitrary connected shape A. Such seed-shapes are very convenient as they are capable of "simulating" the *universal transformation* techniques that are possible if we have both rotation and sliding movements available (discussed in the next paragraph). To this end, we prove that all seeds of size ≤ 4 cannot serve for this purpose, by proving that they cannot even walk the perimeter of a simple line shape. Then we focus on a *6-seed* and prove that such a seed is capable of walking the perimeter of a large family of shapes, called *orthogonally convex* shapes. This is a first indication, that there might be a large family of shapes that can be transformed to each other with rotation only and without breaking connectivity, by extracting a 6-seed and then exploiting it to transfer nodes to the desired positions. To further support this, we prove that the 6-seed is capable of performing such transfers, by detaching pairs of nodes from the shape, attaching them to itself, thus forming an *8-seed* and then being still capable to walk the perimeter of the shape. The complete discussion on these can be found in the full technical report [MSS17b].

Next, we consider the case in which both rotation and sliding are available and insist on connectivity preservation. We first provide a proof that this combination of simple movements is *universal* with respect to (abbreviated "w.r.t." throughout) transformations, as any pair of connected shapes A and B of the same order ("order" of a shape S meaning the number of nodes of S throughout the paper) can be transformed to each other without ever breaking the connectivity throughout the transformation (a first proof of this fact had already appeared in [DP04]). This generic transformation requires $\Theta(n^2)$ sequential movements in the worst case. By a potential-function argument we show that no transformation can improve on this worst-case complexity for some specific pairs of shapes and this lower bound is independent of connectivity preservation; it only depends on the inherent *transformation-distance* between the shapes. To improve on this, either some sort of parallelism must be employed or more powerful movement mechanisms, e.g., movements of whole sub-shapes in one step. We investigate the former approach and prove that there is a *pipelining* general transformation strategy that improves the time to $O(n)$ (parallel time). We also give a matching $\Omega(n)$ lower bound. On the way, we also show that this parallel complexity is feasible even if the nodes are labeled, meaning that individual nodes must end up in specific positions of the target-shape.

Finally, we assume that the nodes are distributed processes equipped with limited local memory and able to perform communicate-compute-move rounds (where, again, both rotation and sliding movements are available) and provide distributed algorithms for a general type of transformation.

In Section 1.2 we discuss further related literature. Section 2 brings together all definitions and basic facts that are used throughout the paper. In Section 3, we study programmable matter systems equipped only with rotation movement. In Section 4, we insist on rotation only, but additionally require that the material maintains connectivity throughout the transformation. In Section 5, we investigate the combined effect of rotation and sliding movements. Finally, in Section 6 we conclude and discuss further research directions that are opened by our work.

1.2. Further Related Work

Mobile and Reconfigurable Robotics. There is a very rich literature on mobile and reconfigurable robotics. In mobile (swarm) robotics systems and models, as are, for example, the models for robot gathering [CFPS03, KKM10] and deployment [SMO+18] (cf., also [FPS12]), geometric pattern formation [SY99, DFSY15], and connectivity preservation [CKLWL09], the modules are usually robots equipped with some mobility mechanism making them free to move in any direction of the plane (and in some cases even continuously). In contrast, we only allow discrete movements relative to neighboring nodes. Modular self-reconfigurable robotic systems form an area on their own, focusing on aspects like the design, motion planning, and control of autonomous robotic modules [BKRT04, YSS+07, ABD+13, YUY16]. The model considered in this paper bears similarities to some of the models that have appeared in this area. The main difference is that we follow a more computation-theoretic approach, while the studies in this area usually follow a more applied perspective.

Puzzles. Puzzles are combinatorial one-player games, usually played on some sort of board. Typical questions of interest are whether a given puzzle is solvable and finding the solution that makes the fewest number of moves. Answers to these questions range from being in **P** up to **PSPACE**-hard or even undecidable when some puzzles are generalized to the entire plane with unboundedly many pieces [Dem01, HD05]. Famous examples of puzzles are the Fifteen Puzzle, Sliding Blocks, Rush Hour, Pushing Blocks, and Solitaire. Even though none of these is equivalent to the model considered here, the techniques that have been developed for solving and characterizing puzzles may turn out to be very useful in the context of programmable matter systems. Actually, in some cases, such puzzles show up as special cases of the transformation problems considered here (e.g., the Fifteen Puzzle may be obtained if we restrict a transformation of node-labeled shapes to take place in a 4x4 square region); such connections of programmable matter systems and models to puzzles have also been recently highlighted in [BDF$^+$17].

Passive Systems. Most of the models discussed so far, including the model under consideration in this paper, are *active* models, meaning that the movements are under the complete control of the algorithm. In contrast, in *passive* models the underlying algorithm cannot control the movements but in most cases it can decide in some way which movements to accept and which not. The typical assumption is that the movements are controlled by a *scheduler* (possibly adversarial), which represents some dynamicity of the system or the environment. Population Protocols [AAD$^+$06, AAER07] and variants are a typical such example. For example, in Network Constructors [MS16] nodes move around randomly due to the dynamicity of the environment and when two of them interact the protocol can decide whether to establish a connection between them; that is, the protocol has some *implicit* control of the system's dynamics. Another passive model, inspired from biological multicellular processes, was recently proposed by Emek and Uitto [EU17]. Most models from the theory of algorithmic self-assembly, like the Abstract Tile Assembly Model (aTAM) [Win98, RW00], fall also in this category. In this paper, we are only concerned with active systems. Hybrid models combining active capabilities and passive dynamics, remain an interesting open research direction. For a review of passive systems and, in general, of the theory of dynamic networks, the interested reader is encouraged to consult [MS18].

Metamorphic Systems. As already mentioned, [DSY04a, DSY04b, DP04] are very close to our approach therefore, despite the fact that they naturally belong to models of mobile and reconfigurable robotics, we discuss them separately. All modeling and results in the present paper were developed independently from the above studies, but as there are some similarities we now compare to those papers. In particular, the model of rotations and slidings on the grid that we studied, turns out to be the same as the model considered in those papers. The study of rotation alone that we explore in Sections 3 and 4 to the best of our knowledge has not been considered in the literature. Regarding rotation and sliding combined, discussed in Section 5, the universality result that we prove turns out to have been first proven in [DP04], but we leave our proof for completeness. Both proofs exploit the idea of converting a given shape A first into a straight line and then to the target shape B, but the proof arguments are different. Regarding [DSY04b], in that paper the authors mainly studied a distributed version of the transformation problems. They also posed a number of decidability questions, some of them proved to be undecidable (we do not deal with undecidable problems in this paper). Their main question regarding decidability problems was related to the universality result later proved in [DP04]. Our distributed results, only briefly mentioned at the end of Section 5 (to be published separately, as the focus of the present paper is feasibility of transformations and centralized transformation algorithms), use a different model than the one studied in [DSY04a, DSY04b]. The main difference is that in [DSY04a, DSY04b] each node can communicate to the whole network in any given round (mostly, though, in the sense that a node can have a complete observation of the current global configuration), while in our case communication is restricted to be local. An interesting idea, explored in [DSY04b] that has also been exploited in our study (but for centralized transformations in our case) is the pipelining technique, by which more than one nodes traverse the perimeter of a shape at the same time in order to speed up transformation. In their case, it is used on the special case of connected shapes in which no row has a gap, i.e., every row is a line of consecutive nodes, while in our case it is used to speed up the universality result to linear parallel

time. Finally, it should be mentioned that some authors have studied alternative geometries than the one considered here, such as representations of systems in which each module is a regular hexagon (see, for instance, [NGY00, WWA04].

2. Preliminaries

The programmable matter systems considered in this paper operate on a 2D square grid, with each position (or *cell*) of the grid being uniquely referred to by its $y \geq 0$ and $x \geq 0$ coordinates. [1] Such a system consists of a set V of n *modules*, called *nodes* throughout. Each node may be viewed as a spherical module fitting inside a cell of the grid. At any given time, each node $u \in V$ occupies a cell $o(u) = (o_y(u), o_x(u)) = (i, j)$ (omitting the time index for simplicity here and also whenever clear from context) and no two nodes may occupy the same cell. [2] In some cases, when a cell is occupied by a node we may refer to that cell by a color, e.g., *black*, and when a cell is not occupied (in which case we may also call it *empty*) we usually color it white. At any given time t, the positioning of nodes on the grid defines an undirected *neighboring relation* $E(t) \subset V \times V$, where $\{u, v\} \in E$ iff $o_y(u) = o_y(v)$ and $|o_x(u) - o_x(v)| = 1$ or $o_x(u) = o_x(v)$ and $|o_y(u) - o_y(v)| = 1$, that is, if u and v are either *horizontal* or *vertical* neighbors on the grid, respectively. It is immediate to observe that every node can have at most 4 neighbors at any given time. A more informative way to define the system at a given time t, and thus often more convenient, is as a mapping $P_t \colon \mathbb{N}_{\geq 0} \times \mathbb{N}_{\geq 0} \to \{0, 1\}$ where $P_t(i, j) = 1$ iff cell (i, j) is occupied by a node.

At any given time t, $P_t^{-1}(1)$ defines a *shape*. [3] Such a shape is called *connected* if $E(t)$ defines a connected graph. A connected shape is called *convex* if for any two occupied cells, the line in $\mathbb{R}^2$ through their centers does not pass through an empty cell. A connected shape A is called *orthogonally convex* if the intersection of any vertical or horizontal line with A is either empty or connected. Equivalently, this means that for any two cells occupied by A, belonging either to the same row or the same column, the line that connects their centers does not pass through an empty cell (i.e., compared to pure convexity, we now exclude diagonal lines). We call a shape *compact* if it has no holes.

In general, shapes can *transform* to other shapes via a sequence of one or more *movements* of individual nodes. Time consists of discrete *steps* (or *rounds*) and in every step, zero or more movements may occur, possibly following a centralized or distributed computation sub-step, depending on the application. In the *sequential* case, at most one movement may occur per step, and in the *parallel* case any number of "valid" movements may occur in parallel. [4] We consider two types of movements: (i) *rotation* and (ii) *sliding*. In both movements, a single node moves relative to one or more neighboring nodes as we now explain.

A single *rotation* movement of a node u is a 90° *rotation* of u around one of its neighbors. Let (i, j) be the current position of u and let its neighbor be v occupying the cell $(i - 1, j)$ (i.e., lying below u). Then u can *rotate* 90° clockwise (counterclockwise) around v iff the cells $(i, j + 1)$ and $(i - 1, j + 1)$ ($(i, j - 1)$ and $(i - 1, j - 1)$, respectively) are both empty. By rotating the whole system 90°, 180°, and 270°, all possible rotation movements are defined analogously. See Figure 1 for an illustration.

A single *sliding* movement of a node u is a one-step horizontal or vertical movement "over" a horizontal or vertical line of (neighboring) nodes of length 2. In particular, if (i, j) is the current position of u, then u can *slide* rightwards to position $(i, j + 1)$ iff $(i, j + 1)$ is not occupied and there exist nodes at positions $(i - 1, j)$ and $(i - 1, j + 1)$ or at positions $(i + 1, j)$ and $(i + 1, j + 1)$, or both. Precisely the same definition holds for up, left, and down sliding movements by rotating the whole system 90°, 180°, and 270° counterclockwise,

[1] We should make clear at this point that the grid assumption is only used to facilitate intuition of the geometric properties of the model (that is, the permissible arrangements of the nodes) and presentation of our results. Therefore, the systems in this work consist of the nodes only and the grid is not considered part of the studied systems.

[2] As these are discrete coordinates, we have preferred to use the (i, j) matrix notation, where i is the row and j the column, instead of the (x, y) Cartesian coordinates, but this doesn't make any difference.

[3] P_t^{-1} denotes the inverse function of P_t, therefore $P_t^{-1}(1)$ returns the set of all cells that are occupied by nodes.

[4] By "valid", we mean here subject to the constraint that their whole movement paths correspond to pairwise disjoint sub-areas of the grid.

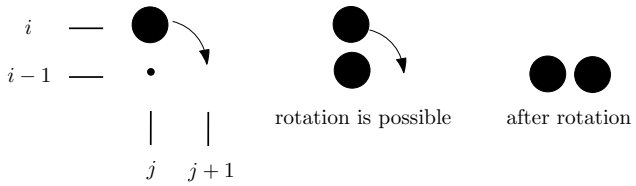

Figure 1: Rotation clockwise. A node on the black dot (in row $i-1$) and empty cells at positions $(i, j+1)$ and $(i-1, j+1)$ are required for this movement. Then an example movement is given.

respectively. Intuitively, a node can slide one step in one direction if there are two consecutive nodes either immediately "below" or immediately "above" that direction that can assist the node slide (see Figure 2). [5]

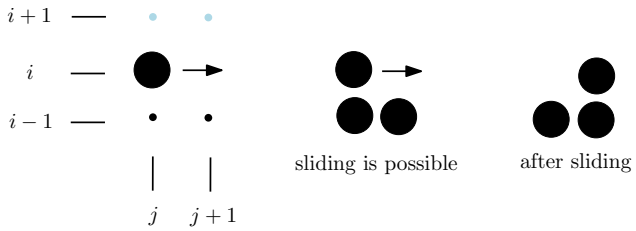

Figure 2: Sliding to the right. Either the two light blues (dots in row $i+1$; appearing gray in print) or the two blacks (dots in row $i-1$) and an empty cell at position $(i, j+1)$ are required for this movement. Then an example movement with the two blacks is given.

Let A and B be two shapes. We say that A *transforms to* B *via a movement* m (which can be either a rotation or a sliding), denoted $A \xrightarrow{m} B$, if there is a node u in A such that if u applies m, then the shape resulting after the movement is B (possibly after rotations and translations of the resulting shape, depending on the application). 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 $A \xrightarrow{m} B$ for some movement m. We say that A *transforms* to B (or that B is *reachable* from A) and write $A \rightsquigarrow B$, if there is a sequence of shapes $A = C_0, C_1, \ldots, C_t = B$, such that $C_i \to C_{i+1}$ for all i, $0 \leq i < t$. We should mention that we do not always allow m to be any of the two possible movements. In particular, in Sections 3 and 4 we only allow m to be a rotation. We shall clearly explain what movements are permitted in each part of the paper. Observe now that both rotation and sliding are *reversible* movements, a fact that we extensively use in our results. Based on this, we may prove that:

Proposition 1. *The relation "transforms to" (i.e., '$\rightsquigarrow$') is a partial equivalence relation.*

Proof. The relation '$\rightsquigarrow$' is a binary relation on shapes. To show that it is a partial equivalence relation, we have to show that it is symmetric and transitive.

For symmetry, we have to show that for all shapes A and B, if $A \rightsquigarrow B$ then $B \rightsquigarrow A$. It suffices to show that for all A, B, if $A \to B$ then $B \to A$, meaning that every one-step transformation (which can be either a single rotation or a single sliding) can be *reversed*. For the rotation case, this follows by observing that a rotation of a node u can be performed iff there are two consecutive empty positions in its trajectory. When u rotates, it leaves its previous position empty, thus, leaving in this way two consecutive positions empty for the reverse rotation to become enabled. The argument for sliding is similar.

For transitivity, we have to show that for all shapes A, B, and C, if $A \rightsquigarrow B$ and $B \rightsquigarrow C$ then $A \rightsquigarrow C$. By definition, $A \rightsquigarrow B$ if there is a sequence of shapes $A = C_0, C_1, \ldots, C_t = B$, such that $C_i \to C_{i+1}$ for all i, $0 \leq i < t$ and $B \rightsquigarrow C$ if there is a sequence of shapes $B = C_t, C_{t+1}, \ldots, C_{t+l} = C$, such that $C_i \to C_{i+1}$ for

[5] Observe that there are plausible variants of the present definition of sliding, such as to slide with nodes at $(i-1, j)$ and $(i+1, j+1)$ or even with a single node at $(i-1, j)$ or at $(i+1, j)$. In this paper, though, we only focus on our original definition.

all i, $t \leq i < t+l$. So, for the sequence $A = C_0, C_1, \ldots, C_t = B, C_{t+1}, \ldots, C_{t+l} = C$ it holds that $C_i \to C_{i+1}$ for all i, $0 \leq i < t+l$, that is, $A \rightsquigarrow C$. □

When the only available movement is rotation, there are shapes in which no rotation can be performed (such examples are provided in Section 3). If we introduce a *null* rotation, then every shape may transform to itself by applying the *null* rotation. That is, reflexivity is also satisfied, and, together with symmetry and transitivity from Proposition 1, '$\rightsquigarrow$' (by rotations only) becomes an equivalence relation.

Definition 1. *Let A be a connected shape. Color black each cell of the grid that is occupied by a node of A. A cell (i, j) is part of a* hole *of A if every infinite length single path starting from (i, j) (moving only horizontally and vertically) necessarily goes through a black cell. Color black also every cell that is part of a hole of A, to obtain a compact black shape A'. Consider now polygons defined by unit-length line segments of the grid. Define the* perimeter *of A as the minimum-area such polygon that completely encloses A' in its interior. The fact that the polygon must have an* interior *and an* exterior *follows directly from the Jordan curve theorem [Jor93].*

Definition 2. *Now, color red any cell of the grid that has contributed at least one of its line segments to the perimeter and is not black (i.e., is not occupied by a node of A). Call this the* cell-perimeter *of shape A. See Figure 3 for an example.*

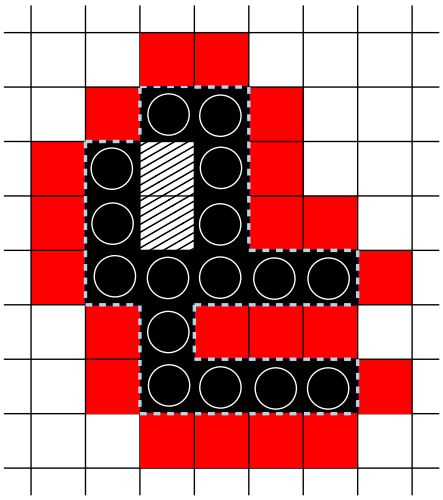

Figure 3: The perimeter (polygon of unit-length dashed line segments colored light blue; appearing dashed gray in print) and the cell-perimeter (cells colored red; appearing gray in print, throughout the paper) of a shape A (white spherical nodes; their corresponding cells have been colored black). The dashed black cells correspond to a hole of A.

Definition 3. *The* 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 perimeter.*

Definition 4. *The* extended external surface *of a connected shape A, is defined by adding to A's external surface all nodes of A whose cell shares a corner with A's perimeter (for example, the black node just below the hole in Figure 3).*

Proposition 2. *The extended external surface of a connected shape A, is itself a connected shape.*

Proof. The perimeter of A is connected, actually, it is a cycle. This connectivity is preserved by the extended external surface, as whenever the perimeter moves straight, we have two horizontally or vertically neighboring nodes on the extended external surface and whenever it makes a turn, we either stay put or preserve connectivity via an intermediate diagonal node (from those nodes used to extend the external surface). □

Observe, though, that the extended external surface is not necessarily a cycle. For example, the extended external surface of a line-shape is equal to the shape itself (and, therefore, a line).

2.1. Problem Definitions

We here provide formal definitions of all the transformation problems that are considered in this work.

ROT-TRANSFORMABILITY**.** Given a connected initial shape A and a connected target shape B, decide whether A can be transformed to B (under translations and rotations of the shapes) using only a sequence of rotation movements.

ROTC-TRANSFORMABILITY**.** The special case of ROT-TRANSFORMABILITY in which A and B are again connected shapes and connectivity must be preserved throughout the transformation.

RS-TRANSFORMABILITY**.** Given a connected initial shape A and a connected target shape B, decide whether A can be transformed to B (under translations and rotations of the shapes) by a sequence of rotation and sliding movements.

Minimum-Seed-Determination. Given an initial shape A and a target shape B (usually only with rotation available and a proof that A and B are not transformable to each other without additional assumptions) determine a minimum-size seed and an initial positioning of that seed relative to A that makes the transformation from A to B feasible. There are several meaningful variations of this problem. For example, the seed may or may not be eventually part of the target shape or the seed may be used as an intermediate step to show feasibility with "external" help and then be able to show that, instead of externally providing it, it is possible to *extract* it from the initial shape A via a sequence of moves. We will clearly indicate which version is considered in each case.

In the above problems, the goal is to show feasibility of a set of transformation instances and, if possible, to provide an algorithm that decides feasibility. [6]

In the last part of the paper, we consider *distributed transformation tasks*. There, the nodes are *distributed processes* able to perform *communicate-compute-move rounds* and the goal is to program them so that they (algorithmically) self-transform their initial arrangement to a target arrangement.

Distributed-Transformability. Given an initial shape A and a target shape B (usually by having access to both rotation and sliding), the nodes (which are now distributed processes), starting from A, must transform themselves to B by a sequence of communication-computation-movement rounds. In the distributed transformations, we mostly consider the case in which A can be *any connected shape* and B is a *spanning line*, i.e., a linear arrangement of all the nodes.

3. Rotation

In this section, the only permitted movement is 90° *rotation* around a neighbor. Our main result in this section is that ROT-TRANSFORMABILITY $\in \mathbf{P}$.

Consider a black and red checkered coloring of the 2D grid, similar to the coloring of a chessboard. Then any shape S may be viewed as a colored shape consisting of $b(S)$ blacks and $r(S)$ reds ($b(S)$ and $r(S)$ denoting the number of black and reds, respectively, of a shape S). Call two shapes A and B *color-consistent* if $b(A) = b(B)$ and $r(A) = r(B)$ and call them *color-inconsistent* otherwise. Call a transformation from a shape A to a shape C *color-preserving* if A and C are color consistent. Observe now, that if $A \to B$, then A and B are color-consistent, because a rotation can never move a node to a position of different color

[6] An immediate next goal is to devise an algorithm able to compute an actual transformation or even compute or approximate the *optimum* transformation (usually w.r.t. the number of moves). We leave these as interesting open problems.

than its starting position. This implies that if $A \rightsquigarrow C$, then A and C are color-consistent, because any two consecutive shapes in the sequence are color-consistent. We conclude that:

Observation 1. *The rotation movement is color-preserving. Formally, $A \rightsquigarrow C$ (restricted to rotation only) implies that A and C are color-consistent. In particular, every node beginning from a black (red) position of the grid, will always be on black (red, respectively) positions throughout a transformation consisting only of rotations.*

Based on this property of the rotation movement, we may call each node *black* or *red* throughout a transformation, based only on its initial coloring. Observation 1 gives a partial way to determine that two shapes A and B cannot be transformed to each other by rotations.

Proposition 3. *If two shapes A and B are color-inconsistent, then it is impossible to transform one to the other by rotations only.*

We now show that two shapes being color-consistent does not necessarily mean that they can be transformed to each other by rotations. We begin with a proposition relating the number of black and red nodes in a connected shape.

Proposition 4. *A connected shape with k blacks has at least $\lceil (k-1)/3 \rceil$ and at most $3k+1$ reds.*

Proof. For the upper bound, observe that a black can hold up to 4 distinct reds in its neighborhood, which implies that k blacks can hold up to $4k$ reds in total, even if the blacks were not required to be connected to each other. To satisfy connectivity, every black must share a red with some other black (if a black does not satisfy this, then it cannot be connected to any other black). Any such sharing reduces the number of reds by at least 1. As at least $k-1$ such sharings are required for each black to participate in a sharing, it follows that we cannot avoid a reduction of at least $k-1$ in the number of reds, which leaves us with at most $4k-(k-1) = 3k+1$ reds.

For the lower bound, if we invert the roles of blacks and reds, we have that l reds can hold at most $3l+1$ blacks. So, if k is the number of blacks, it holds that $k \leq 3l+1 \Leftrightarrow l \geq (k-1)/3$ and due to the fact that the number of reds must be an integer, we conclude that for k blacks the number of reds must be at least $\lceil (k-1)/3 \rceil$. □

Proposition 5. *There is a generic connected shape, called* line-with-leaves*, that has a color-consistent version for any connected shape. In other words, for k blacks it covers the whole range of reds from $\lceil (k-1)/3 \rceil$ to $3k+1$ reds.*

Proof. Let red be the majority color of A and k be the number of black nodes of A. Consider a line of alternating red and black nodes, starting and ending with a black node, such that all k black nodes are exhausted (see Figure 4). To do this, $k-1$ reds are needed in order to alternate blacks and reds on the line. Next, "saturate" every black (i.e. maximize its degree) by adding as many red nodes as it can fit around it (recall that the maximum degree of every node is 4). The resulting saturated shape has k blacks and $3k+1$ reds. This shape covers the $3k+1$ upper bound on the possible number of reds. By removing red leaf-nodes (i.e., of degree 1) one after the other, we can achieve the whole range of numbers of reds, from $k-1$ to $3k+1$ reds. It suffices to restrict attention to the range from k to $3k+1$ reds. Take now any connected shape A and color it in such a way that red is the majority color, that is $l \geq k$, where l is the number of reds and k is the number of blacks (there is always a way to do that). From the upper bound of Proposition 4, l can be at most $3k+1$, so we have $k \leq l \leq 3k+1$ for any connected shape A, which falls within the range that the line-with-leaves can represent. Therefore, we conclude that any connected shape A has a color-consistent shape B from the line-with-leaves family. □

Based on this, we now show that the inverse of Proposition 3 is not true, that is, it does not hold that any two color-consistent shapes can be transformed to each other by rotations.

Proposition 6. *There is an infinite set of pairs (A, B) of connected shapes, such that A and B are color-consistent but cannot be transformed to each other by rotations only.*

Figure 4: A saturated line-with-leaves shape, in which there are $k = 5$ blacks and $3k + 1 = 16$ reds.

Proof. For shape A, take a rhombus as shown in Figure 5, consisting of k^2 blacks and $(k+1)^2$ reds, for any $k \geq 2$. In this shape, every black node is "saturated", meaning that it has 4 neighbors, all of them necessarily red. This immediately excludes the blacks from being able to move, as all their neighboring positions are occupied by reds. But the same holds for the reds, as all potential target-positions for a rotation are occupied by reds. Thus, no rotation movement can be applied to any such shape A and A can only be transformed to itself (by *null* rotations). By Proposition 5, any such A has a color-consistent shape B from the family of line-with-leaves shapes, such that $B \neq A$ (actually in B several blacks may have degree 3 in contrast to A where all blacks have degree 4). We conclude that A and B are distinct color-consistent shapes which cannot be transformed to each other, and there is an infinite number of such pairs, as the number k^2 of black nodes of A can be made arbitrarily large. □

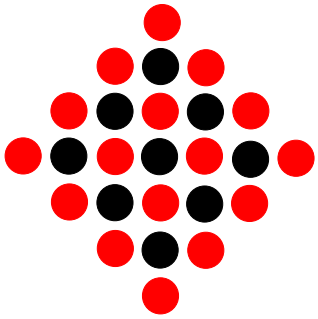

Figure 5: A rhombus shape, consisting of $k^2 = 9$ blacks and $(k+1)^2 = 16$ reds.

Propositions 3 and 6 give a partial characterization of pairs of shapes that cannot be transformed to each other. Observe that the impossibilities proved so far hold for all possible transformations based on rotation only, including those that are allowed to break connectivity.

A small shape of particular interest is a bi-color pair or *2-line*. Such pairs can move easily in any direction, which makes them very useful components of transformations. One way to simplify some transformations would be to identify as many such pairs as possible in a shape and treat them in a different way than the rest of the nodes. A question in this respect is whether all the minority-color nodes of a connected shape can be completely matched to (distinct) nodes of the majority color. We show that this is not true.

Proposition 7. *There is an infinite family of connected shapes such that if A is a shape of size n in the family, then any matching of A leaves at least $n/8$ nodes of each color unmatched.*

Proof. The counterexample is given in Figure 6. The shape consists of a square and a number of flowers emanating from it. A red flower is one with three red petals and a black centre and inversely for black flowers. Observe first that any two consecutive flowers on square cannot be of the same color. We can partition the shape into sub-shapes consisting of 6 nodes of the square and 2 flowers, i.e., 16 nodes in total (such as the top left sub-shape highlighted in Figure 6). Then it is sufficient to observe that any at least two petals of any flower cannot get matched to a neighbor of opposite color, therefore 2 red nodes for the red flower and 2 black nodes from the black flower will remain unmatched, which corresponds to 1/8 of the nodes of the sub-shape will be unmatched reds and another 1/8 will be unmatched blacks. □

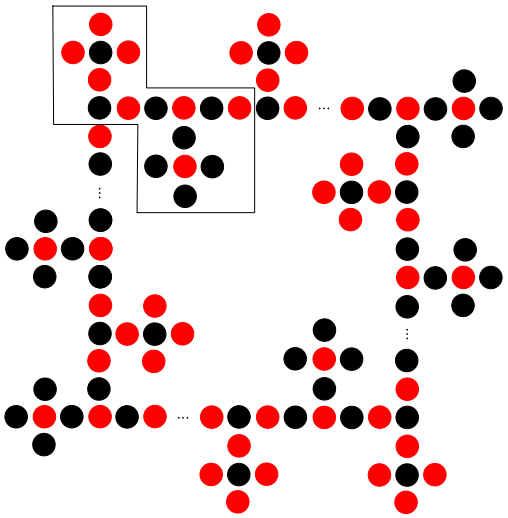

Figure 6: The counterexample.

Recall that Rot-Transformability is the language of all transformation problems between connected shapes that can be solved by rotation only and RotC-Transformability is its subset obtained by the restriction that the transformation should not break the connectivity of the shape at any point during the transformation. We begin by showing that the inclusion between the two languages is strict, that is, there are strictly more feasible transformations if we allow connectivity to break. We prove this by showing that there is a feasible transformation, namely folding a spanning line in half, in Rot-Transformability\RotC-Transformability.

Theorem 1. RotC-Transformability $\subset$ Rot-Transformability.

Proof. RotC-Transformability $\subseteq$ Rot-Transformability is immediate, as any transformation using only rotations that does not break the shape's connectivity is also a valid transformation for Rot-Transformability. So, it suffices to prove that there is a transformation problem in Rot-Transformability\RotC-Transformability. Consider a (connected) horizontal line of any even length n, and let $u_1, u_2, \ldots, u_n$ be its nodes. The transformation asks to fold the line onto itself, forming a double-line of length $n/2$ and width 2, i.e., a $n/2 \times 2$ rectangle.

It is easy to observe that this problem is not in RotC-Transformability for any $n > 4$: the only nodes that can rotate without breaking connectivity are u_1 and u_n, but any of their two possible rotations only enables a rotation that will bring the nodes back to their original positions. This means that, if the transformation is not allowed to break connectivity, then such a shape is trapped in a loop in which only the endpoints can rotate between three possible positions, therefore it is impossible to fold a line of length greater than 4.

On the other hand, if connectivity can be broken, we can perform the transformation by the following simple procedure, consisting of $n/4$ phases: In the beginning of every phase $i \in \{1, 2, \ldots, \lfloor n/4 \rfloor\}$, pick the nodes u_{2i-1}, u_{2i}, which shall at that point be the two leftmost nodes of the original line. Rotate u_{2i-1} once clockwise, to move above u_{2i}, then u_{2i} three times clockwise to move to the right of u_{2i-1} (the first of these three rotations breaks connectivity and the third restores it), and then rotate u_{2i-1} twice clockwise to move to the right of u_{2i}, then u_{2i} twice clockwise to move to the right of u_{2i-1} and repeat this alternation until the pair that moves to the right meets the previous pair, which will be when u_{2i-1} becomes the left neighbor of u_{2i-2} on the upper line of the rectangle under formation, or, in case $i = 1$, when u_{2i-1} goes above u_n (see Figure 7). If $n/4$ is not an integer, then perform a final phase, in which the leftmost node of the original line is rotated once clockwise to move above its right neighbor, and this completes folding. □

This means that allowing the connectivity to break enables more transformations, and this motivates us

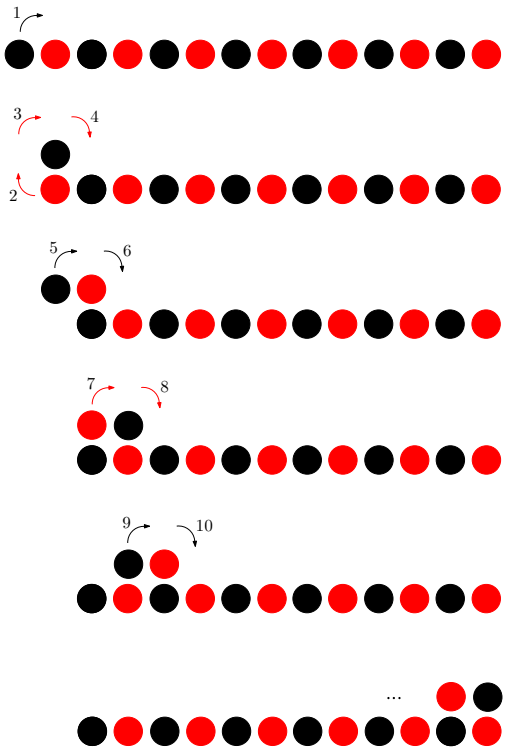

Figure 7: Line folding.

to start from this simpler case. But we already know from Proposition 6 that even in this case an infinite number of pairs of shapes cannot be transformed to each other. Aiming at a general transformation, we ask whether there is some minimal addition to a shape that would allow it to transform. The solution turns out to be as small as a *2-line seed* (a bi-color pair, usually referred to as "2-line" or "2-seed") lying initially somewhere "outside" the boundaries of the shape (e.g., just below the lowest row occupied by the shape).

Based on the above assumptions, we shall now prove that any pair of color-consistent connected shapes A and B can be transformed to each other. Recall from the discussion before Proposition 7 that 2-line shapes can move freely in any direction. The idea is to exploit the fact that the 2-line can move freely in any direction and to use it in order to extract from A another 2-line. In this way, a 4-line seed is formed, which can also move freely in all directions. Then we use the 4-line as a transportation medium for carrying the nodes of A, one at a time. We exploit these mobility mechanisms to transform A into a uniquely defined shape from the line-with-leaves family of Proposition 5. But if any connected shape A with an extra 2-line can be transformed to its color-consistent line-with-leaves version with an extra 2-line, then this also holds inversely due to reversibility, and it follows that any A can be transformed to any B by transforming A to its line-with-leaves version L_A and then inverting the transformation from B to $L_B = L_A$.

Theorem 2. *If connectivity can break and there is a 2-line seed provided "outside" the initial shape, then any pair of color-consistent connected shapes A and B can be transformed to each other by rotations only.*

Proof. Without loss of generality (abbreviated "w.l.o.g." throughout), due to symmetry and the 2-line's unrestricted mobility, it suffices to assume that the seed is provided somewhere below the lowest row l occupied by the shape A. We show how A can be transformed to L_A with the help of the seed. We define L_A as follows: Let k be the cardinality of the minority color, let it be the black color. As there are at least k reds, we can create a horizontal line of length $2k$, i.e., $u_1, u_2, \ldots, u_{2k}$, starting with a black (i.e., u_1 is black), and alternating blacks and reds. In this way, the blacks are exhausted. The remaining $\leq (3k+1)-k = 2k+1$ reds are then added as leaves of the black nodes, starting from the position to the left of u_1 and continuing

counterclockwise, i.e., below u_1, below u_3, ..., below u_{2k-1}, above u_{2k-1}, above u_{2k-3}, and so on. This gives the same shape from the line-with-leaves family, for all color-consistent shapes (observe that the leaf to the right of the line is always placed). L_A shall be constructed on rows $l-5$ to $l-3$ (not necessarily inclusive), with u_1 on row $l-4$ and a column j preferably between those that contain A.

First, extract a 2-line from A, from row l, so that the 2-line seed becomes a 4-line seed. To see that this is possible for every shape A of order at least 2, distinguish the following two cases: (i) If the lowest row has a horizontal 2-line, then the 2-line can leave the shape without any help and approach the 2-seed. (ii) If not, then take any node u of row l. As A is connected and has at least two nodes, u must have a neighbor v above it. The only possibility that the 2-line u,v is not free to leave A is when v has both a left and a right neighbor. Figure 8 shows how this can be resolved with the help of the 2-line seed (now the 2-line seed approaches and extracts the 2-line).

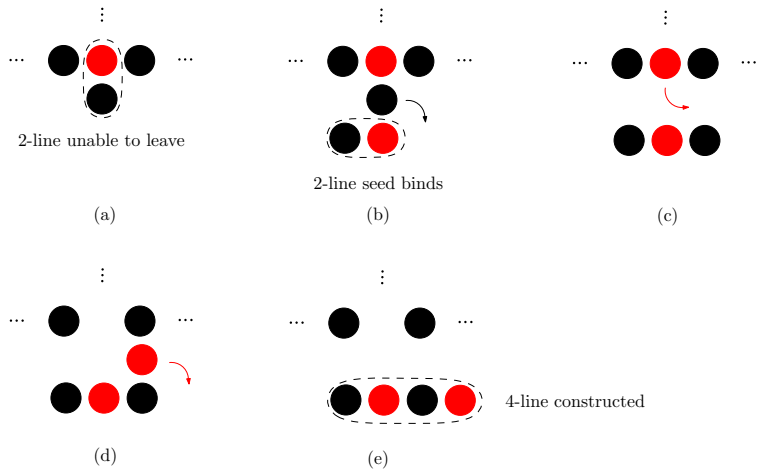

Figure 8: Extracting a 2-line with the help of the 2-line seed.

To transform A to L_A, given the 4-line seed, do the following:

- While black is still present in A:
  - If on the current lowest row occupied by A, there is a 2-line that can be extracted alone and moved towards L_A, then perform the shortest such movement that attaches the 2-line to the right endpoint of L_A's line $u_1, u_2, \ldots$.
  - If not, then do the following. Maintain a *repository* of nodes at the empty space below row $l-7$, initially empty. If, either in the lowest row of A or in the repository, there is a node of opposite color than the current color of the right endpoint of L_A's line, use the 4-line to transfer such a node and make it the new right endpoint of L_A's line. Otherwise, use the 4-line to transfer a node of the lowest row of A to the repository.
- Once black has been exhausted from A and the repository (i.e., when u_{2k-3} has been placed; u_{2k-1} and u_{2k} will only be placed in the end as they are part of the 4-line), transfer a red to position u_{2k-2}. If there are no more nodes left, run the termination phase, otherwise transfer the remaining nodes (all red) with the 4-line, one after the other, and attach them as leaves around the blacks of L_A's line, beginning from the position to the left of u_1 counterclockwise, as described above (skipping position u_{2k}).
- Termination phase: the line-with-leaves is ready, apart from positions u_{2k-1}, u_{2k} which require a 2-line from the 4-line. If the position above u_{2k-1} is empty, then extract a 2-line from the 4-line and transfer it to the positions u_{2k-1}, u_{2k}. This completes the transformation. If the position above u_{2k-1} is occupied by a node u_{2k+1}, then place the whole 4-line vertically with its lowest endpoint on u_{2k} (as

in Figure 9). Then rotate the top endpoint counterclockwise to move above u_{2k+1}, then rotate u_{2k+1} clockwise around it to move to its left, then rotate the node above u_{2k} counterclockwise to move to u_{2k-1}, and finally restore u_{2k+1} to its original position. This completes the construction (the 2-line that always remains can be transferred in the end to a predetermined position).

□

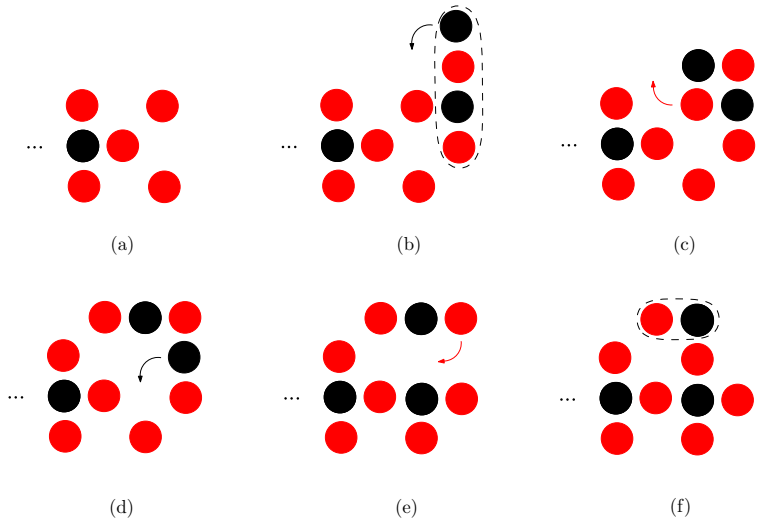

Figure 9: The termination phase of the transformation.

The natural next question is to what extent the 2-line seed assumption can be dropped. Clearly, by Proposition 6, this cannot be always possible. The following corollary gives a sufficient condition to drop the 2-line seed assumption, without looking deep into the structure of the shapes that satisfy it.

Corollary 1. *Assume rotations only and that connectivity can break. Let A and B be two color-consistent connected shapes such that each one of them can self-extract a 2-line. Then A and B can be transformed to each other.*

We remind that a rotation move in a grid can occur towards four directions: $NorthEast(1)$, $SouthEast(2)$, $SouthWest(3)$, $NorthWest(4)$. In order for the first move to occur, a node has to be present North OR East but not both, and similarly for the other directions. If the connectivity of the shape can be broken and two nodes, u and v, are next to each other and u can perform a rotation using v, then v can perform a rotation using u if the connectivity of the shape can be broken.

We now arrive at a sufficient and necessary condition for dropping the 2-line seed assumption.

Lemma 1. *A 2-seed can be extracted from a shape A iff at least one node of A can rotate.*

Proof. If no node of A can rotate, then no movement is possible and trivially a 2-seed cannot be extracted from the shape. Therefore it suffices to prove that if any node of A can rotate then a 2-seed can be extracted from A (meaning that after a sequence of permissible movements a 2-seed will end up at the exterior of A, i.e., outside its perimeter). We distinguish two cases:

- *Case 1: There is a node u incident to the perimeter of A that can move.* In this case, there must be at least two neighboring nodes u, v at positions $(i, j), (i, j+1)$ such that there are no nodes at position $(i+1, j), (i+1, j+1)$ or in another symmetric orientation. In any of these cases, the 2-seed consists of the nodes u, v and can readily be extracted from shape A. The only exception is when the empty space is part of a tunnel of width 1. In this case, by focusing to the nodes adjacent to the outermost two empty cells of the tunnel, these two nodes can be extracted as a 2-seed.

- *Case 2: There is an internal node u (i.e., not incident to the perimeter) of A that can move. In this case, w.l.o.g. (because all other cases are symmetric to the one considered) there must be two consecutive empty cells $(i,j),(i,j+1)$ with the two cells $(i-1,j),(i-1,j+1)$ below them being occupied by nodes. It is straightforward to reconfigure these four cells by two permissible rotations, so that cells $(i-1,j),(i-1,j+1)$ become now empty and cells $(i,j),(i,j+1)$ occupied. By induction, as long as the cells below are occupied we can continue shifting the two consecutive empty cells downwards. If we now focus on the downmost 2 such empty cells irrespective of their orientation (the other directions are symmetric)* then shifting them downwards until the perimeter is reached is always possible as the space between those cells and the perimeter is filled with nodes apart possibly from isolated empty cells (i.e., an empty cell surrounded by nodes). In all these cases downward shifting can continue until the perimeter is reached, and when that happens a movement becomes available at the perimeter which is covered by the previous case.

□

Theorem 3. Rot-Transformability $\in \mathbf{P}$.

Proof. In Lemma 1, we proved that we can extract a 2-seed from a shape iff a move is initially available. By Theorem 2, if both shapes A and B have a 2-seed available then they can be transformed to each other. It follows that two shapes A and B can be transformed to each other iff both have a move available. If yes, *accept*, otherwise, *reject*. These checks can be easily performed in polynomial time as follows. Consider a $n \times n$ grid, in which any shape with n nodes can fit. The time it takes for an algorithm to check if one of the shapes has a move available is $O(n)$. If for example the algorithm checks each individual node, that takes $O(1)$ time and, therefore, $O(n)$ time for n nodes. So for two shapes it takes $O(n)$ time to check if a move is available in each of the shapes. Thus, the problem belongs to $\mathbf{P}$. □

4. Rotation and Connectivity Preservation

In this section, we restrict our attention to transformations that transform a connected shape A into one of its color-consistent shapes B using only rotations without ever breaking the connectivity of the shape on the way. As already mentioned in the introduction, connectivity preservation is a very desirable property for programmable matter, as, among other positive implications, it guarantees that communication between all nodes is maintained, it minimizes transformation failures, requires less sophisticated actuation mechanisms, and increases the external forces required to break the system apart.

We begin by proving that RotC-Transformability can be decided in deterministic polynomial space.

Theorem 4. RotC-Transformability $\in$ **PSPACE**.

Proof. We first present a *nondeterministic* Turing machine (NTM) N that decides RotC-Transformability in polynomial space. N takes as input two shapes A and B, both consisting of n nodes and at most $4n$ edges. A reasonable representation is in the form of a binary $n \times n$ matrix (representing a large enough sub-area of the grid) where an entry is 1 iff the corresponding position is occupied by a node. Given the present configuration C, where $C = A$ initially, N nondeterministically picks a valid rotation movement of a single node. This gives a new configuration C'. Then N replaces the previous configuration with C' in its memory, by setting $C \leftarrow C'$. Moreover, N maintains a counter *moves* (counting the number of moves performed so far), with maximum value equal to the total number of possible shape configurations, which is at most 2^{n^2} in the binary matrix encoding of configurations. To set up such a counter, N just have to reserve for it n^2 (binary) tape-cells, all initialized to 0. Every time N makes a move, as above, after setting a value to C' it also increases *moves* by 1, i.e., sets $moves \leftarrow moves + 1$. Then N takes another move and repeats. If it ever holds that $C' = B$ (may require N to perform a polynomial-space pattern matching on the $n \times n$ matrix to find out), then N accepts. If it ever holds that the counter is exhausted, that is, all its bits are set to 1, N rejects. If A can be transformed to B, then there must be a transformation beginning from A and producing B, by a sequence of valid rotations, without ever repeating a shape. Thus, some branch of

N's computation will follow such a sequence and accept, while all non-accepting branches will reject after at most 2^{n^2} moves (when *moves* reaches its maximum value). If A cannot be transformed to B, then all branches will reject after at most 2^{n^2} moves. Thus, N correctly decides RotC-Transformability. Every branch of N, at any time, stores at most two shapes (the previous and the current), which requires $O(n^2)$ space in the matrix representation, and a 2^{n^2}-counter which requires $O(n^2)$ bits. It follows that every branch uses space polynomial in the size of the input. So, far we have proved that RotC-Transformability is decidable in nondeterministic polynomial (actually, linear) space. By applying Savitch's theorem [Sav70] [7], we conclude that RotC-Transformability is also decidable in deterministic polynomial space (actually, quadratic), i.e., it is in **PSPACE**. □

Recall that in the line folding problem, the initial shape is a (connected) horizontal line of any even length n, with nodes $u_1, u_2, \ldots, u_n$, and the transformation asks to fold the line onto itself, forming a double-line of length $n/2$ and width 2. As part of the proof of Theorem 1, it was shown that if $n > 4$, then it is impossible to solve the problem by rotation only (if $n = 4$, it is trivially solved, just by rotating each endpoint above its unique neighbor). In the next proposition, we employ again the idea of a seed to show that with a little external help the transformation becomes feasible.

Proposition 8. *If there is a 3-line seed v_1, v_2, v_3, horizontally aligned over nodes u_3, u_4, u_5 of the line, then the line can be folded without breaking connectivity.*

Proof. We distinguish two cases, depending on whether we want the seed to be part of the final folded line or not. If yes, then we can either use a 4-line seed directly, over nodes u_3, u_4, u_5, u_6, or a 3-line seed but require n to be odd (so that $n + 3$ is even). If not, then n must be even. We show the transformation for the first case, with n odd and a 3-line seed (the other cases can be then treated with minor modifications).

We first show a simple reduction from an odd line with a 3-line seed starting over its third node to an even line with a 4-line seed starting over its third node. By rotating u_1 clockwise over u_2, we obtain the 4-line seed u_1, v_1, v_2, v_3. It only remains to move the whole seed two positions to the right (by rotating each of its 2-lines clockwise around themselves). In this manner, we obtain an even-length line $u_2, \ldots, u_n$ and a 4-line seed starting over its third node, without breaking connectivity. Therefore, in what follows we may assume w.l.o.g. (due to the described reduction) that the initial shape is an even-length line $u_1, u_2, \ldots, u_n$ with a 4-line seed v_1, v_2, v_3, v_4 horizontally aligned over nodes u_3, u_4, u_5, u_6. Now, Figure 10 gives a step-by-step procedure (call it a *phase*) to exploit this 4-line seed in order to remove two nodes from the leftmost part of the line (e.g., nodes u_1 and u_2 in Figure 10) and add them to the line to be formed over the original line (i.e., initially two nodes shall be transferred over the rightmost part of the line; nodes v_1 and v_4 in Figure 10). By the end of a phase, the left part of the shape is identical in structure to the original one, having the 4-line seed at exactly the same position relative to the line as before, therefore the same procedure can be repeated over and over until the line has been completely folded. □

As already shown in Theorem 1, the connectivity-preservation constraint increases the class of infeasible transformations. As highlighted in Proposition 8, a convenient turnaround in such cases could be to introduce a suitable seed that can assist the transformation. So, for instance, in Proposition 8 we circumvented the impossibility of folding a line $u_1, u_2, \ldots, u_n$ in half, by adding a 3-line seed v_1, v_2, v_3, horizontally aligned over nodes u_3, u_4, u_5 of the line. Interestingly, adding the seed over nodes u_4, u_5, u_6 does not work. Therefore, given an infeasible transformation, a natural next question is to find a minimum seed (could be any connected small shape, not necessarily a line) and a placement of that seed that enables the transformation (*Minimum-Seed-Determination* problem). In the following theorem (whose proof can be found in the full technical report [MSS17b]), we try to identify a minimum seed that can walk the perimeter of any shape. We leave as an interesting open problem whether such a shape is able to move nodes gradually to a predetermined

[7] Informally, Savitch's theorem establishes that any NTM that uses $f(n)$ space can be converted to a deterministic TM that uses only $f^2(n)$ space. Formally, it establishes that for any function $f\colon \mathbb{N} \to \mathbb{N}$, where $f(n) \geq \log n$, $\mathbf{NSPACE}(f(n)) \subseteq \mathbf{SPACE}(f^2(n))$.

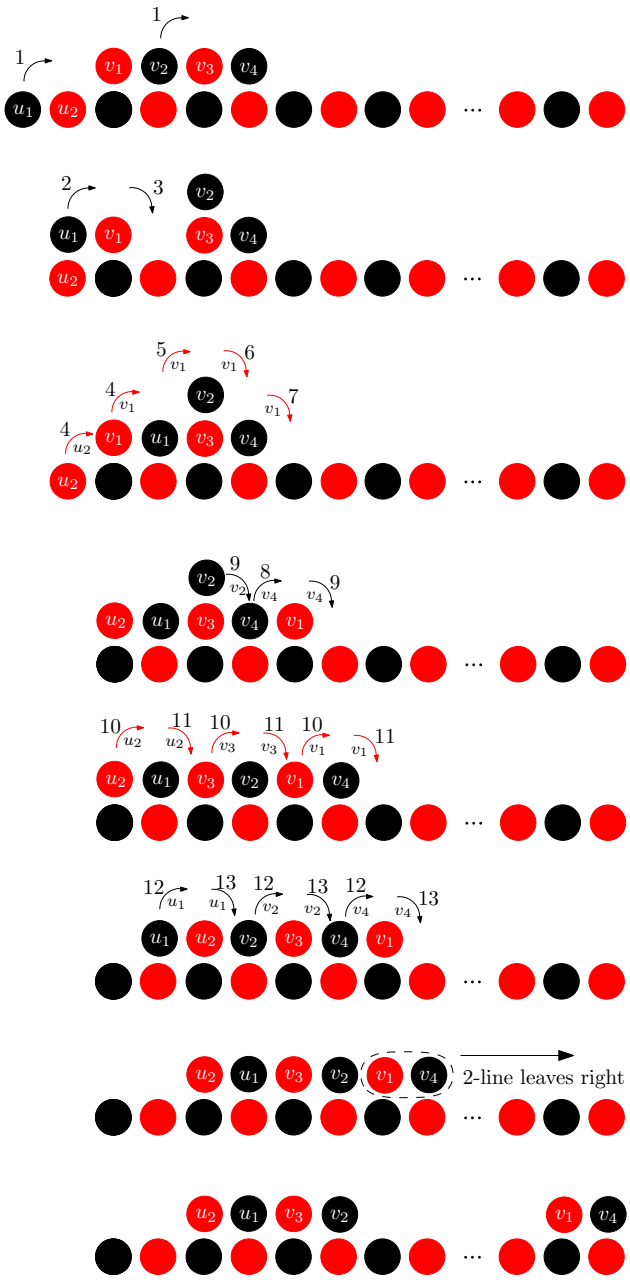

Figure 10: The main subroutine (i.e., a phase) of line folding with connectivity preservation.

position, in order to transform the initial shape into a line-with-leaves (as in Theorem 2, but without ever breaking connectivity this time). [8]

Theorem 5. *If connectivity must be preserved: (i) Any* (≤ 4)*-seed cannot traverse the perimeter of a line, (ii) A 6-seed can traverse the perimeter of any orthogonally convex shape.*

5. Rotation and Sliding

In this section, we study the combined effect of rotation and sliding movements. We begin by proving (in Theorem 6) that rotation and sliding together are *transformation-universal*, meaning that they can transform any given shape to any other shape of the same size without ever breaking the connectivity during the transformation. It would be useful for the reader to recall Definitions 1, 2, 3, and 4 and Proposition 2, from Section 2, as the results that follow make extensive use of them.

As the perimeter is a (connected) polygon, it can be traversed by a "particle" walking on its edges (the unit-length segments). We call it a particle as an imaginary entity that can walk through the edges of the grid and distinguish from nodes of the shape that may move form cell to cell. We now show how to "simulate" the particle's movement and traverse the cell-perimeter by a node, using rotation and sliding only. If needed, the reader is encouraged to refresh notions like "line segment" and "perimeter" by looking at Figure 3 in Section 2. Additionally, staying consistent with Figure 3, we partition the cells of the grid into red, which are the cells of the cell-perimeter, and black, which are the cells occupied by the shape.

Lemma 2. *If we place a node u on any position of the cell-perimeter of a connected shape A, then u can walk the whole cell-perimeter and return to its original position by using only rotations and slidings.*

Proof. We show how to "simulate" the walk of a particle moving on the edges of the perimeter. The simulation implements the following simple rules:

1. If both the current line segment traversed by the particle and the line segment traversed in the previous step correspond to edges of the same red cell, then stay put.
2. If not:
   (a) If the two consecutive line segments traversed form a line segment of length 2, then move by sliding one position in the same direction as the particle.
   (b) If the two consecutive line segments traversed are perpendicular to each other, then move by a single rotation in the same direction as the particle. [9]

It remains to prove that u can indeed always perform the claimed movements. (1) is trivial. The node temporarily stops at a "corner" and always moves in the next step according to condition 2. For (2.a), a line segment of length 2 on the perimeter is always defined by two consecutive blacks to the interior and two consecutive empty cells to the exterior (belonging to the cell-perimeter), therefore, u can slide on the empty cells. For (2.b), there must be a black in the internal angle defined by the line segments (because, by Definition 1 we have colored black all cells in the interior of the perimeter, that is the shape itself plus its holes if any) and an empty cell diagonally to it, in the exterior (for an example, see the right black node on the highest row containing nodes of A, in Figure 3, Section 2). Therefore, rotation can be performed. □

Next, we shall prove that u need not be an additional node, but actually a node belonging to the shape, and in particular one of those lying on the shape's boundary.

Lemma 3. *Let A be a connected shape of order at least 2. Then there is a subset R of the nodes on A's external surface, such that $|R| \geq 2$ and for all $u \in R$, if we completely remove u from A, then the resulting shape $A' = A - \{u\}$ is also connected.*

[8] Another way to view this, is as an attempt to simulate the universal transformations based on combined rotation and sliding (presented in Section 5), in which single nodes are able to walk the perimeter of the shape.

[9] Observe that when these perpendicular line segments correspond to edges of the same red cell, then the node will not rotate but will instead stay put due to prior execution of rule 1.

Proof. If the extended external surface of A contains a cycle, then such a cycle must necessarily have length at least 4 (due to geometry). In this case, any node of the intersection of the external surface (non-extended) and the cycle can be removed without breaking A's connectivity. If the extended external surface of A does not contain a cycle, then it corresponds to a tree graph which by definition has at least 2 leaves, i.e., nodes of degree exactly 1. Any such leaf can be removed without breaking A's connectivity. In both cases, $|R| \geq 2$. □

Lemma 4. *Pick any $u \in R$ (R defined on a connected shape A as above). Then u can walk the whole cell-perimeter of $A' = A - \{u\}$ by rotations and slidings.*

Proof. It suffices to observe that u already lies on the cell-perimeter of A'. Then, by Lemma 2, it follows that such a walk is possible. □

We are now ready to state and prove the universality theorem of rotations and slidings. As already mentioned, this result was first proved in [DP04], but we here give our independently developed proof.

Theorem 6 ([DP04])**.** *Let A and B be any connected shapes, such that $|A| = |B| = n$. Then A and B can be transformed to each other by rotations and slidings, without breaking the connectivity during the transformation.*

Proof. It suffices to show that any connected shape A can be transformed to a spanning line L using only rotations and slidings and without breaking connectivity during the transformation. If we show this, then A can be transformed to L and B can be transformed to L (as A and B have the same order, therefore correspond to the same spanning line L), and by reversibility of these movements, A and B can be transformed to each other via L.

Pick the rightmost column of the grid containing at least one node of A, and consider the lowest node of A in that column. Call that node u. Observe that all cells to the right of u are empty. Let the cell of u be (i, j). The final constructed line will start at (i, j) and end at $(i, j + n - 1)$.

The transformation is partitioned into $n - 1$ phases. In each phase k, we pick a node from the original shape and move it to position $(i, j + k)$, that is, to the right of the right endpoint of the line formed so far. In phase 1, position $(i, j + 1)$ is a cell of the cell-perimeter of A. So, even if it happens that u is a node of degree 1, by Lemma 3, there must be another such node $v \in A$ that can walk the whole cell-perimeter of $A' = A - \{v\}$ (the latter, due to Lemma 4). As $u \neq v$, $(i, j + 1)$ is also part of the cell-perimeter of A', therefore, v can move to $(i, j+1)$ by rotations and slidings. As A' is connected (by Lemma 3), $A' \cup \{(i, j+1)\}$ is also connected and also all intermediate shapes were connected, because v moved on the cell-perimeter and, therefore, it never disconnected from the rest of the shape during its movement.

In general, the transformation preserves the following invariant. At the beginning of phase k, $1 \leq k \leq n - 1$, there is a connected shape $S(k)$ (where $S(1) = A$) to the left of column j (j inclusive) and a line of length $k - 1$ starting from position $(i, j + 1)$ and growing to the right. Restricting attention to $S(k)$, there is always a $v \neq u$ that could (hypothetically) move to position $(i, j + 1)$ if it were not occupied. This implies that before the final movement that would place v on $(i, j+1)$, v must have been in $(i+1, j)$ or $(i+1, j+1)$, if we assume that v always walks in the clockwise direction. Observe now that from each of these positions v can perform zero or more right slidings above the line in order to reach the position above the right endpoint $(i, j + k - 1)$ of the line. When this occurs, a final clockwise rotation makes v the new right endpoint of the line. The only exception is when v is on $(i + 1, j + 1)$ and there is no line to the right of (i, j) (this implies the existence of a node on $(i + 1, j)$, otherwise connectivity of $S(k)$ would have been violated). In this case, v just performs a single downward sliding to become the right endpoint of the line. □

Theorem 7. *The transformation of Theorem 6 requires $\Theta(n^2)$ movements in the worst case.*

Proof. A *staircase* is defined as a shape of the form $(i, j), (i-1, j), (i-1, j+1), (i-2, j+1), (i-2, j+2), (i-3, j+2), \ldots$. Consider such a staircase shape of order n, as depicted in Figure 11. The strategy of Theorem 6 will choose to construct the line to the right of node u. The only node that can be selected to move in each phase without breaking the shape's connectivity is the top-left node. Initially, this is v, which must

perform $\lceil n/2 \rceil$ movements to reach its position to the right of u. In general, the total number of movements M, performed by the transformation of Theorem 6 on the staircase, is given by

$$\begin{aligned} M &= \left\lceil \frac{n}{2} \right\rceil + 2 \cdot \sum_{i=1}^{(n-3)/2} \left(\left\lceil \frac{n}{2} \right\rceil + i \right) \\ &= \left\lceil \frac{n}{2} \right\rceil (n-2) + 2 \cdot \sum_{i=1}^{(n-3)/2} i \\ &= \Theta(n^2). \end{aligned}$$

□

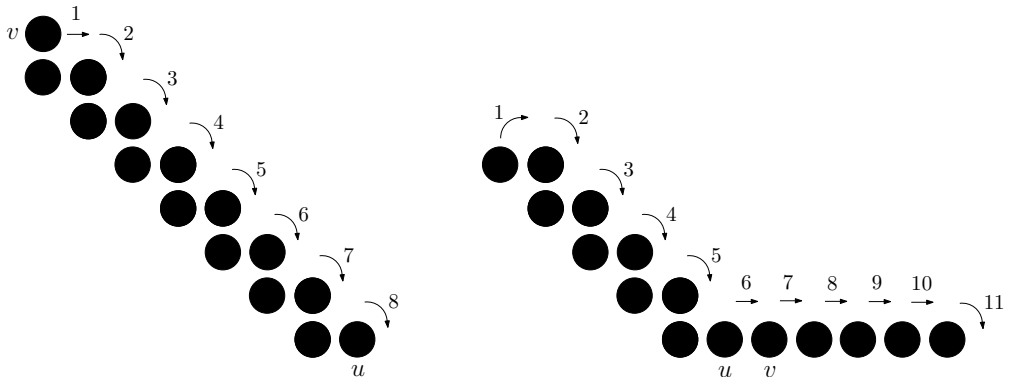

Figure 11: Transforming a staircase into a spanning line.

Theorem 7 shows that the above generic strategy is slow in some cases, as is the case of transforming a staircase shape into a spanning line. We shall now show that there are pairs of shapes for which any strategy and not only this particular one, may require a quadratic number of steps to transform one shape to the other.

Definition 5. *Define the* potential of a shape A *as its minimum "distance" from the line L, where $|A| = |L|$. The* distance *is defined as follows: Consider any placement of L relative to A and any pairing of the nodes of A to the nodes of the line. Then sum up the Manhattan distances* [10] *between the nodes of each pair. The minimum sum between all possible relative placements and all possible pairings is the distance between A and L and also A's potential.* [11] *In case the two shapes do not have an equal number of nodes, then any matching is not perfect and the distance can be defined as infinite.*

Observe that the potential of the line is 0 as it can be totally aligned on itself and the sum of the distances is 0.

Lemma 5. *The potential of the staircase is $\Theta(n^2)$.*

Proof. We prove it for horizontal placement of the line, as the vertical case is symmetric. Any such placement leaves either above or below it at least half of the nodes of the staircase (maybe minus 1). W.l.o.g. let it be above it. Every two nodes, the height increases by 1, therefore there are 2 nodes at distance 1, 2 at distance

[10] The Manhattan distance between two points (i, j) and (i', j') is given by $|i - i'| + |j - j'|$.

[11] To make this constructive, it is sufficient to restrict the possible placements of the line to those were at least one node of the line overlaps with the shape. Note that even without this, the potential is finite and well defined as it corresponds to a placement that minimizes the above nonnegative sum.

2,..., 2 at distance n/4. Any matching between these nodes and the nodes of the line gives for every pair a distance at least as large as the vertical distance between the staircase's node and the line, thus, the total distance is at least $2 \cdot 1 + 2 \cdot 2 + ... + 2 \cdot (n/4) = 2 \cdot (1 + 2 + ... + n/4) = (n/4) \cdot (n/4 + 1) = \Theta(n^2)$. We conclude that the potential of the staircase is $\Theta(n^2)$. □

Theorem 8. *Any transformation strategy based on rotations and slidings which performs a single movement per step requires $\Theta(n^2)$ steps to transform a staircase into a line.*

Proof. To show that $\Omega(n^2)$ movements are needed to transform the staircase into a line, it suffices to observe that the difference in their potentials is that much and that one rotation or one sliding can decrease the potential by at most 1. □

Remark 1. *The above lower bound is independent of connectivity preservation. It is just a matter of the total distance based on single distance-one movements.*

Finally, it is interesting to observe that such lower bounds can be computed in polynomial time, because there is a polynomial-time algorithm for computing the distance between two shapes.

Proposition 9. *Let A and B be connected shapes. Then their distance $d(A, B)$ can be computed in polynomial time.*

Proof. The algorithm picks a node $u \in B$, a cell c of the grid occupied by a node $v \in A$, and an orientation $o \in north, east, south, west$ and draws a copy of the shape B, starting with u on c and respecting the orientation o. Then, it constructs (in its memory) a complete weighted bipartite graph (X, Y), where X and Y are equal to the node-sets of A and B, respectively. The weight $w(x, y)$ for $x \in X$ and $y \in Y$ is defined as the distance from x to y (given the drawing of shape B relative to shape A). To compute the minimum total distance pairing of the nodes of A and B for this particular placement of A and B, the algorithm computes a minimum cost perfect matching of (X, Y), e.g., by the Kuhn-Munkres algorithm (a.k.a. the Hungarian algorithm) [Kuh55], and the sum k of the weights of its edges, and sets $dist = \min\{dist, k\}$, initially $dist = \infty$. Then the algorithm repeats for the next selection of $u \in B$, cell c occupied by a node $v \in A$, and orientation o. In the end, the algorithm gives $dist$ as output. To see that $dist = d(A, B)$, observe that the algorithm just implements the procedure for computing the distance, of Definition 5, with the only differences being that it does not check all pairings of the nodes, instead directly computes the minimum-cost pairing, and that it does not try all relative placements of A and B but only those in which A and B share at least one cell of the grid. To see that this selection is w.l.o.g., assume that a placement of A and B in which no cell is shared achieves the minimum distance and observe that, in this case, A could be shifted one step "closer" to B, strictly decreasing their distance and, thus, contradicting the optimality of such a placement. As the different possible relative placements of A and B are $4n^2$ (place each node of B on every node of A and for each such placement there are 4 possible orientations) and the Kuhn-Munkres algorithm is a polynomial-time algorithm (in the size of the bipartite graph), we conclude that the algorithm computes the distance in polynomial time. □

To give a faster transformation either pipelining must be used (allowing for more than one movement in parallel) or more complex mechanisms that move sub-shapes consisting of many nodes, in a single step. In what remains, we follow the former approach by allowing an unbounded number of rotation and/or sliding movements to occur simultaneously in a single step (though, in pairwise disjoint areas).

5.1. Parallelizing the Transformations

We now maintain the connectivity preservation requirement but allow an unbounded number of rotation and/or sliding movements to occur simultaneously in a single step.

Proposition 10. *There is a pipelining strategy that transforms a staircase into a line in $O(n)$ parallel time.*

Proof. Number the nodes of the staircase 1 through n starting from the top and following the staircase's connectivity until the bottom-right node is reached. This gives an odd-numbered upper diagonal and an even-numbered lower diagonal. Node 1 moves as in Theorem 6. Any even node w starts moving as long as its upper odd neighbor has reached the same level as w (e.g., node 2 first moves after node 1 has arrived to the right of node 3). Any odd node $z > 1$ starts moving as long as its even left neighbor has moved one level down (e.g., node 3 first moves after node 2 has arrived to the right of 5). After a node starts moving, it moves in every step as in Theorem 6 (but now many nodes can move in parallel, implementing a pipelining strategy). It can be immediately observed that any node i starts after at most 3 movements of node $i-1$ (actually, only 2 movements for even i), so after roughly at most $3n$ steps, node $n-2$ starts. Moreover, a node that starts, arrives at the right endpoint of the line after at most n steps, which means that after at most $4n = O(n)$ steps all nodes have taken their final position in the line. □

Proposition 10 gives a hint that pipelining could be a general strategy to speed-up transformations. We next show how to generalize this technique to any possible pair of shapes.

Theorem 9. *Let A and B be any connected shapes, such that $|A| = |B| = n$. Then there is a pipelining strategy that can transform A to B (and inversely) by rotations and slidings, without breaking the connectivity during the transformation, in $O(n)$ parallel time.*

Proof. The transformation is a pipelined version of the sequential transformation of Theorem 6. Now, instead of picking an arbitrary next candidate node of $S(k)$ to walk the cell-perimeter of $S(k)$ clockwise, we always pick the rightmost clockwise node $v_k \in S(k)$, that is, the node that has to walk the shortest clockwise distance to arrive at the line being formed. This implies that the subsequent candidate node v_{k+1} to walk, is always "behind" v_k in the clockwise direction and is either already free to move or is enabled after v_k's departure. Observe that after at most 3 clockwise movements, v_k can no longer be blocking v_{k+1} on the (possibly updated) cell-perimeter. Moreover, the clockwise move of v_{k+1} only introduces a gap in its original position, therefore it only affects the structure of the cell-perimeter "behind" it. The strategy is to start the walk of node v_{k+1} as soon as v_k is no longer blocking its way. As in Proposition 10, once a node starts, it moves in every step, and again any node arrives at the end of the line being formed after at most n movements. It follows, that if the pipelined movement of nodes cannot be blocked in any way, after $4n = O(n)$ steps all nodes must have arrived at their final positions. Observe now that the only case in which pipelining could be blocked is when a node is sliding through a (necessarily dead-end) "tunnel" of height 1 (such an example is the red tunnel on the third row from the bottom, in Figure 3 of Section 2). To avoid this, the nodes shortcut the tunnel, by visiting only its first position (i, j) and then simply skipping the whole walk inside it (that walk would just return them to position (i, j) after a number of steps). □

We next show that even if A and B are labeled shapes, that is, their nodes are assigned the indices $1, \ldots, n$ (uniquely, i.e., without repetitions), we can still transform the labeled A to the labeled B with only a linear increase in parallel time. We only consider transformations in which the nodes never change indices in any way (e.g., cannot transfer them, or swap them), so that each particular node of A must eventually occupy (physically) a particular position of B (the one corresponding to its index).

Corollary 2. *The labeled version of the transformation of Theorem 9 can be performed in $O(n)$ parallel time.*

Proof. Recall from Theorem 6 that the line was constructed to the right of some node u. That node was the lowest node in that column, therefore, there is no node below u in that column. The procedure of Theorem 9, if applied on the labeled versions of A and B will result in two (possibly differently) labeled lines, corresponding to two permutations of $1, 2, \ldots, n$, call them π_A and π_B. It suffices to show a way to transform π_A to π_B in linear parallel time, as then labeled A is transformed to π_A, then π_A to π_B, and then π_B to B (by reversing the transformation from B to π_B), all in linear parallel time.

To do this, we actually slightly modify the procedure of Theorem 9, so that it does not construct π_A in the form of a line, but in a different form that will allow us to quickly transform it to π_B without breaking connectivity. What we will construct is a double line, with the upper part growing to the right of node u as

before and the lower part starting from the position just below u and also growing to the right. The upper line is an unordered version of the left half of π_B and the lower line is an unordered version of the right half of π_B. To implement the modification, when a node arrives above u, as before, if it belongs to the upper line, it goes to the right endpoint of the line as before, while if it belongs to the lower line, it continues its walk in order to reach the right endpoint of the lower line.

When the transformation of labeled A to the folded line is over, the procedure has to order the nodes of the folded line and then unfold in order to produce π_B. We first order the upper line in ascending order. While we do this, the lower line stays still in order to preserve the connectivity. When we are done, we order the lower line in descending order, now keeping the upper line still. Finally, we perform a parallel right sliding of the lower line (requiring linear parallel time), so that its inverse permutation ends up to the right of the upper line, thus forming π.

It remains to show how the ordering of the upper line can be done in linear parallel time without breaking connectivity. To do this, we simulate a version of the odd-even sort algorithm (a.k.a. parallel bubble sort) which sorts a list of n numbers with $O(n)$ processors in $O(n)$ parallel time. The algorithm progresses in odd and even phases. In the odd phases, the odd positions are compared to their right neighbor and in the even phases to their left neighbor and if two neighbors are ever found not to respect the ordering a swap of their values is performed. In our simulation, we break each phase into two subphases as follows. Instead of performing all comparisons at once, as we cannot do this and preserve connectivity, in the first subphase we do every second of them and in the second subphase the rest so that between any pair of nodes being compared there are 2 nodes that are not being compared at the same time. Now if the comparison between the i-th and the $(i+1)$-th node indicates a swap, then $i+1$ rotates over $i+2$, i slides right to occupy the previous position of $i+1$, and finally $i+1$ slides left over i and then rotates left around i to occupy i's previous position. This swapping need 4 steps and does not break connectivity. The upper part has $n/2$ nodes, each subphase takes 4 steps to swap everyone (in parallel), each phase has 2 sub-phases, and $O(n)$ phases are required for the ordering to complete, therefore, the total parallel time is $O(n)$ for the upper part and similarly $O(n)$ for the lower part. This completes the proof. □

An immediate observation is that a linear-time transformation does not seem satisfactory for all pairs of shapes. To this end, take a square S and rotate its top-left corner u one position clockwise, to obtain an almost-square S'. Even though, a single counter-clockwise rotation of u suffices to transform S' to S, the transformation of Theorem 9 may go all the way around and first transform S' into a line and then transform the line to S. In this particular example, the distance between S and S', according to Definition 5, is 2, while the generic transformation requires $\Theta(n)$ parallel time. So, it is plausible to ask if any transformation between two shapes A and B can be performed in time that grows as a function of their distance $d(A, B)$. We show that this cannot always be the case, by presenting two shapes A and B with $d(A, B) = 2$, such that A and B require $\Omega(n)$ parallel time to be transformed to each other.

Proposition 11. *There are two shapes A and B with $d(A, B) = 2$, such that A and B require $\Omega(n)$ parallel time to be transformed to each other.*

Proof. The two shapes, a black and a red one, are depicted in Figure 12. Both shapes form a square which is empty inside and also open close to the middle of its bottom side. The difference between the two shapes is the positioning of the bottom "door" of length 2. The red shape has it exactly in the middle of the side, while the black shape has it shifted one position to the left. Equivalently, the bottom side of the red shape is "balanced", meaning that it has an equal number of nodes in each side of the vertical dashed axis that passes through the middle of the bottom, while the black shape is "unbalanced" having one more node to the right of the vertical axis than to its left.

To transform the black shape into the red one, a black node must necessary cross either the vertical or the horizontal axis (e.g., the black node u to move all the way around and end up at the same cell as the red node v). Because, if nothing of the two happens, then, no matter the transformation, we won't be able to place the axes so that the running shape has two pairs of balanced quadrants, while, on the other hand, the red shape satisfies this, by pairing together the two bottom quadrants and the two upper quadrants. Clearly, no move can be performed in the upper quadrants initially, as this would break the shape's connectivity. The

only black nodes that can move initially are u and w and no other node can ever move unless first approached by some other node that could already move. Observe also that u and w cannot cross the vertical boundary of their quadrants, unless with help of other nodes. But the only way for a second node to move in any of these quadrants (without breaking connectivity) is for either u or w to reach the corner of their quadrant which takes at least $n/8 - 2$ steps and then another $n/8$ steps for any (or both) of these nodes to reach the boundary, that is, at least $n/4 - 2$ steps, which already proves the required $\Omega(n)$ parallel-time lower bound (even a parallel algorithm has to pay the initial sequential movement of either u or w). □

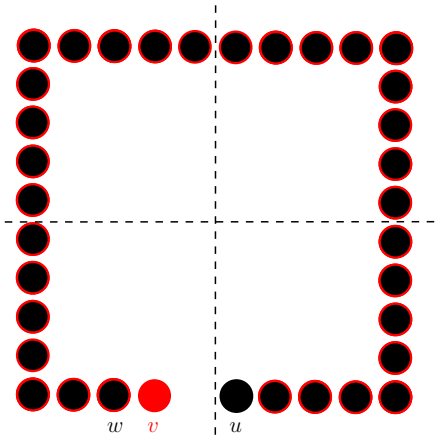

Figure 12: Counterexample for distance.

In the full technical report [MSS17b], we also study the RS-Transformability problem in distributed systems and give an algorithm that transforms a large family of shapes into a spanning line:

Theorem 10. *We provide an algorithm, called* Compact Line*, that can transform any compact shape into a spanning line.*

6. Conclusions and Further Research

There are many open problems related to the findings of the present work. We here restricted attention to the two extremes, in which the transformation either preserves connectivity or is free to break it arbitrarily. A compromise could be to allow some restricted degree of connectivity breaking, like necessarily restoring it in at most $k \geq 0$ steps (a special case of this had been already proposed as an open question in [DP04]). There are other meaningful "good" properties that we would like to maintain throughout a transformation. An interesting example is the *strength* of the shape. One of the various plausible definitions is as the minimum strength sub-shape of the shape (i.e., its weakest part; could possibly be captured by some sort of minimum geometric cuts). Then, a strength-preserving transformation would be one that reaches the target shape while trying to maximize this minimum.

In the transformations considered in this paper, there was no *a priori* constraint on the maximum area that a transformation is allowed to cover or on the maximum dimensions that its intermediate shapes are allowed to have. It seems in general harder to achieve a particular transformation if any of these restrictions is imposed. For example, the generic transformation of Theorem 2 requires some additional space below the shape and the transformations of Theorems 6 and 9 transform any shape first to a spanning line, whose maximum dimension is n, even though the original shape could have a maximum dimension as small as $\sqrt{n}$. Another interesting fact about restricting the boundaries is that in this way we get models equivalent to several interesting puzzles. For example, if the nodes are labeled, the initial shape is a square with a single empty cell, and the boundaries are restricted to the dimensions of the square, we get a generalization of the famous 15-puzzle (see, e.g., [Dem01] for a very nice exposition of this and many more puzzles and 2-player

games). Techniques developed in the context of puzzles could prove valuable for analyzing and characterizing discrete programmable matter systems.

We intentionally restricted attention to very minimal actuation mechanisms, namely rotation and sliding. More sophisticated mechanical operations would enable a larger set of transformations and possibly also reduce the time complexity. Such an example is the ability of a node to become inserted between two neighboring nodes (while pushing them towards opposite directions). This could enable parallel mergings of two lines of length $n/2$ into a line of length n in a single step (an, thus, for example, transforming a square to a line in polylogarithmic time). Another, is the capability of rotating whole lines of nodes (like rotating arms, see, e.g., [WCG+13]). The geometry of the individual modules seems to be a parameter that greatly affects the transformation capabilities of the system as well as the algorithmic solutions. It would be interesting to extend the existing studies and problems to 3-dimensional settings (even motivated by real systems such as the Catoms3D system [PB18]).

There are also some promising specific technical questions. We do not yet know what is the complexity of RotC-Transformability. The fact that a 6-seed is capable of transferring pairs of nodes to desired positions suggests that shapes having such a seed in their exterior or being capable of self-extracting such a seed will possibly be able to transform to each other. Even if this turns out to be true, it is totally unclear whether transformations involving at least one of the rest of the shapes are feasible.

Moreover, we didn't study the problem of computing or approximating the optimum transformation. It seems that the problem is computationally hard. A possible approach to prove **NP**-hardness would be by proving **NP**-hardness of Rectilinear Graphic TSP (could be via a reduction from Rectilinear Steiner Tree or Rectilinear TSP, which are both known to be **NP**-complete [GGJ76]) and then giving a reduction from that problem to the problem of a 2-seed exploring a set of locations on the grid.

Finally, regarding the distributed transformations, there are various interesting variations of the model considered here, that would make sense. For example, to assume nodes that are oblivious w.r.t. their orientation or to consider alternative communication principles, such as visibility and asynchronous communication.

Acknowledgements. We would like to thank the anonymous reviewers of this article and of its preliminary version. Their thorough reading and comments have helped us to improve our work substantially. Moreover, we would like to thank the "Algorithmic Foundations of Programmable Matter" seminar [FRRS16] for inspiring us to start working on this type of problems.

References

[AAD+06] D. Angluin, J. Aspnes, Z. Diamadi, M. J. Fischer, and R. Peralta. Computation in networks of passively mobile finite-state sensors. *Distributed Computing*, 18[4]:235–253, March 2006.

[AAER07] D. Angluin, J. Aspnes, D. Eisenstat, and E. Ruppert. The computational power of population protocols. *Distributed Computing*, 20[4]:279–304, November 2007.

[ABD+13] G. Aloupis, N. Benbernou, M. Damian, E. D. Demaine, R. Flatland, J. Iacono, and S. Wuhrer. Efficient reconfiguration of lattice-based modular robots. *Computational geometry*, 46[8]:917–928, 2013.

[BDF+17] A. T. Becker, E. D. Demaine, S. P. Fekete, J. Lonsford, and R. Morris-Wright. Particle computation: complexity, algorithms, and logic. *Natural Computing*, pages 1–21, 2017.

[BKRT04] Z. Butler, K. Kotay, D. Rus, and K. Tomita. Generic decentralized control for lattice-based self-reconfigurable robots. *The International Journal of Robotics Research*, 23[9]:919–937, 2004.

[CFPS03] M. Cieliebak, P. Flocchini, G. Prencipe, and N. Santoro. Solving the robots gathering problem. In *International Colloquium on Automata, Languages, and Programming*, pages 1181–1196. Springer, 2003.

[CKLWL09] A. Cornejo, F. Kuhn, R. Ley-Wild, and N. Lynch. Keeping mobile robot swarms connected. In *Proceedings of the 23rd international conference on Distributed computing*, DISC'09, pages 496–511, Berlin, Heidelberg, 2009. Springer-Verlag.

[CLS+11] X. Chen, L. Li, X. Sun, Y. Liu, B. Luo, C. Wang, Y. Bao, H. Xu, and H. Peng. Magnetochromatic polydiacetylene by incorporation of fe3o4 nanoparticles. *Angewandte Chemie International Edition*, 50[24]:5486–5489, 2011.

[DDG+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 *Proceedings of the 26th ACM symposium on Parallelism in algorithms and architectures (SPAA)*, pages 220–222, 2014.

[DDG+18] J. J. Daymude, Z. Derakhshandeh, R. Gmyr, A. Porter, A. W. Richa, C. Scheideler, and T. Strothmann. On the runtime of universal coating for programmable matter. *Natural Computing*, 17[1]:81–96, 2018.

[Dem01] E. D. Demaine. Playing games with algorithms: Algorithmic combinatorial game theory. In *International Symposium on Mathematical Foundations of Computer Science*, pages 18–33. Springer, 2001.

[DFSY15] S. Das, P. Flocchini, N. Santoro, and M. Yamashita. Forming sequences of geometric patterns with oblivious mobile robots. *Distributed Computing*, 28[2]:131–145, April 2015.

[DGR+15] Z. Derakhshandeh, R. Gmyr, A. W. Richa, C. Scheideler, and T. Strothmann. An algorithmic framework for shape formation problems in self-organizing particle systems. In *Proceedings of the Second Annual International Conference on Nanoscale Computing and Communication*, page 21. ACM, 2015.

[DGR+16] Z. Derakhshandeh, R. Gmyr, A. W. Richa, C. Scheideler, and T. Strothmann. Universal shape formation for programmable matter. In *Proceedings of the 28th ACM Symposium on Parallelism in Algorithms and Architectures*, pages 289–299. ACM, 2016.

[Dot12] D. Doty. Theory of algorithmic self-assembly. *Communications of the ACM*, 55:78–88, 2012.

[Dot14] D. Doty. Timing in chemical reaction networks. In *Proc. of the 25th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 772–784, 2014.

[DP04] A. Dumitrescu and J. Pach. Pushing squares around. In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 116–123. ACM, 2004.

[DSY04a] A. Dumitrescu, I. Suzuki, and M. Yamashita. Formations for fast locomotion of metamorphic robotic systems. *The International Journal of Robotics Research*, 23[6]:583–593, 2004.

[DSY04b] A. Dumitrescu, I. Suzuki, and M. Yamashita. Motion planning for metamorphic systems: Feasibility, decidability, and distributed reconfiguration. *IEEE Transactions on Robotics and Automation*, 20[3]:409–418, 2004.

[EU17] Y. Emek and J. Uitto. Dynamic networks of finite state machines. *Theoretical Computer Science*, 2017.

[FPS12] P. Flocchini, G. Prencipe, and N. Santoro. Distributed computing by oblivious mobile robots. *Synthesis Lectures on Distributed Computing Theory*, 3[2]:1–185, 2012.

[FRRS16] S. Fekete, A. W. Richa, K. Römer, and C. Scheideler. Algorithmic foundations of programmable matter (Dagstuhl Seminar 16271). In *Dagstuhl Reports*, volume 6. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016. Also in *ACM SIGACT News*, 48.2:87-94, 2017.

[GGJ76] M. R. Garey, R. L. Graham, and D. S. Johnson. Some NP-complete geometric problems. In *Proceedings of the eighth annual ACM symposium on Theory of computing*, pages 10–22. ACM, 1976.

[GKR10] K. Gilpin, A. Knaian, and D. Rus. Robot pebbles: One centimeter modules for programmable matter through self-disassembly. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 2485–2492. IEEE, 2010.

[HD05] R. A. Hearn and E. D. Demaine. PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation. *Theoretical Computer Science*, 343[1-2]:72–96, 2005.

[Jor93] C. Jordan. *Cours d'analyse de l'École polytechnique*, volume 1. Gauthier-Villars et fils, 1893.

[KCL+12] A. N. Knaian, K. C. Cheung, M. B. Lobovsky, A. J. Oines, P. Schmidt-Neilsen, and N. A. Gershenfeld. The milli-motein: A self-folding chain of programmable matter with a one centimeter module pitch. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1447–1453. IEEE, 2012.

[KKM10] E. Kranakis, D. Krizanc, and E. Markou. The mobile agent rendezvous problem in the ring. *Synthesis Lectures on Distributed Computing Theory*, 1[1]:1–122, 2010.

[Kuh55] H. W. Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2[1-2]:83–97, 1955.

[LFS+17] G. A. D. Luna, P. Flocchini, N. Santoro, G. Viglietta, and Y. Yamauchi. Brief Announcement: Shape Formation by Programmable Particles. In *Proceedings of the 31st International Symposium on Distributed Computing (DISC)*, volume 91, pages 48:1–48:3, 2017.

[LYS+01] Y. Lu, Y. Yang, A. Sellinger, M. Lu, J. Huang, H. Fan, R. Haddad, G. Lopez, A. R. Burns, D. Y. Sasaki, et al. Self-assembly of mesoscopically ordered chromatic polydiacetylene/silica nanocomposites. *Nature*, 410[6831]:913–917, 2001.

[MS16] O. Michail and P. G. Spirakis. Simple and efficient local codes for distributed stable network construction. *Distributed Computing*, 29[3]:207–237, 2016.

[MS18] O. Michail and P. G. Spirakis. Elements of the theory of dynamic networks. *Communications of the ACM*, 61[2], 2018.

[MSS17a] O. Michail, G. Skretas, and P. 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.

[MSS17b] O. Michail, G. Skretas, and P. G. Spirakis. On the transformation capability of feasible mechanisms for programmable matter. *arXiv preprint arXiv:1703.04381*, 2017.

[NGY00] A. Nguyen, L. J. Guibas, and M. Yim. Controlled module density helps reconfiguration planning. In *Proc. of 4th International Workshop on Algorithmic Foundations of Robotics*, pages 23–36, 2000.

[PB18] B. Piranda and J. Bourgeois. Designing a quasi-spherical module for a huge modular robot to create programmable matter. *Autonomous Robots*, 42:1619–1633, 2018.

[RCN14] M. Rubenstein, A. Cornejo, and R. Nagpal. Programmable self-assembly in a thousand-robot swarm. *Science*, 345[6198]:795–799, 2014.

[Rot06] P. W. Rothemund. Folding dna to create nanoscale shapes and patterns. *Nature*, 440[7082]:297–302, 2006.

[RW00] P. W. K. Rothemund and E. Winfree. The program-size complexity of self-assembled squares. In *Proceedings of the 32nd annual ACM symposium on Theory of computing (STOC)*, pages 459–468, 2000.

[Sav70] W. J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4[2]:177–192, 1970.

[SCWB08] D. Soloveichik, M. Cook, E. Winfree, and J. Bruck. Computation with finite stochastic chemical reaction networks. *Natural Computing*, 7[4]:615–633, 2008.

[SMO+18] M. Shibata, T. Mega, F. Ooshita, H. Kakugawa, and T. Masuzawa. Uniform deployment of mobile agents in asynchronous rings. *Journal of Parallel and Distributed Computing*, 119:92–106, 2018.

[SY99] I. Suzuki and M. Yamashita. Distributed anonymous mobile robots: Formation of geometric patterns. *SIAM J. Comput.*, 28[4]:1347–1363, March 1999.

[WCG+13] 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 *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, pages 353–354. ACM, 2013.

[Win98] E. Winfree. *Algorithmic Self-Assembly of DNA*. PhD thesis, California Institute of Technology, June 1998.

[WWA04] J. E. Walter, J. L. Welch, and N. M. Amato. Distributed reconfiguration of metamorphic robot chains. *Distributed Computing*, 17[2]:171–189, 2004.

[YSS+07] M. Yim, W.-M. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins, and G. S. Chirikjian. Modular self-reconfigurable robot systems [grand challenges of robotics]. *IEEE Robotics & Automation Magazine*, 14[1]:43–52, 2007.

[YUY16] Y. Yamauchi, T. Uehara, and M. Yamashita. Brief announcement: pattern formation problem for synchronous mobile robots in the three dimensional euclidean space. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing*, pages 447–449. ACM, 2016.