# Pushing Lines Helps: Efficient Universal Centralised Transformations for Programmable Matter

Abdullah Almethen, Othon Michail, and Igor Potapov

Department of Computer Science, University of Liverpool, UK
Email: A.almethen@liverpool.ac.uk, Othon.Michail@liverpool.ac.uk, Potapov@liverpool.ac.uk

**Abstract.** In this paper, we study a discrete system of entities residing on a two-dimensional square grid. Each entity is modelled as a node occupying a distinct cell of the grid. The set of all $n$ nodes forms initially a connected shape $A$. Entities are equipped with a linear-strength pushing mechanism that can push a whole line of entities, from 1 to $n$, in parallel in a single time-step. A target connected shape $B$ is also provided and the goal is to *transform* $A$ into $B$ via a sequence of line movements. Existing models based on local movement of individual nodes, such as rotating or sliding a single node, can be shown to be special cases of the present model, therefore their (inefficient, $\Theta(n^2)$) *universal transformations* carry over. Our main goal is to investigate whether the parallelism inherent in this new type of movement can be exploited for efficient, i.e., sub-quadratic worst-case, transformations. As a first step towards this, we restrict attention solely to centralised transformations and leave the distributed case as a direction for future research. Our results are positive. By focusing on the apparently hard instance of transforming a diagonal $A$ into a straight line $B$, we first obtain transformations of time $O(n\sqrt{n})$ without and with preserving the connectivity of the shape throughout the transformation. Then, we further improve by providing two $O(n \log n)$-time transformations for this problem. By building upon these ideas, we first manage to develop an $O(n\sqrt{n})$-time universal transformation. Our main result is then an $O(n \log n)$-time universal transformation. We leave as an interesting open problem a suspected $\Omega(n \log n)$-time lower bound.

**The full version of the paper with all omitted details is available on arXiv at:** https://arxiv.org/abs/1904.12777.

## 1 Introduction

As a result of recent advances in components such as micro-sensors, electromechanical actuators, and micro-controllers, a number of interesting systems are now within reach. A prominent type of such systems concerns collections of small robotic entities. Each individual robot is equipped with a number of actuation/sensing/communication/computation components that provide it with

some autonomy; for instance, the ability to move locally and to communicate with neighbouring robots. Still, individual local dynamics are uninteresting, and individual computations are restricted due to limited computational power, resources, and knowledge. What makes these systems interesting is the collective complexity of the population of devices. A number of fascinating recent developments in this direction have demonstrated the feasibility and potential of such collective robotic systems, where the scale can range from milli/micro [BG15, GKR10, KCL+12, RCN14, YSS+07] down to nano [DDL+09, Rot06].

This progress has motivated the parallel development of a theory of such systems. It has been already highlighted [MS18] that a formal theory (including modelling, algorithms, and computability/complexity) is necessary for further progress in systems. This is because theory can accurately predict the most promising designs, suggest new ways to optimise them, by identifying the crucial parameters and the interplay between them, and provide with those (centralised or distributed) algorithmic solutions that are best suited for each given design and task, coupled with provable guarantees on their performance. As a result, a number of sub-areas of theoretical computer science have emerged such as mobile and reconfigurable robotics [ABD+13, BKRT04, CFPS12, CKLWL09, DFSY15, DDG+18, DGMRP06, DDG+14, DGR+15, DGR+16, DLFS+19, DLFP+18, FPS12, KKM10, MSS19, SMO+18, YS10, YUY16, YSS+07], passively-mobile systems [AAD+06, AAER07, MS16, MS18] including the theory of DNA self-assembly [Dot12, RW00, Win98, WCG+13], and metamorphic systems [DP04, DSY04a, DSY04b, NGY00, WWA04]; connections are even evident with the theory of puzzles [BDF+19, Dem01, HD05]. A latest ongoing effort is to join these theoretical forces and developments within the emerging area of "Algorithmic Foundations of Programmable Matter" [FRRS16]. *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*.

In this paper, we embark from the model studied in [DP04, DSY04a, DSY04b, MSS19], in which a number of spherical devices are given in the form of a (typically connected) shape $A$ lying on a two-dimensional square grid, and the goal is to transform $A$ into a desired target shape $B$ via a sequence of valid movements of individual devices. In those papers, the considered mechanisms were the ability to rotate and slide a device over neighbouring devices (always through empty space). We here consider an alternative (linear-strength) mechanism, by which a line of one or more devices can translate by one position in a single time-step.

As our main goal is to determine whether the new movement under consideration can *in principle* be exploited for sub-quadratic worst-case transformations, we naturally restrict our attention to centralised transformations. We generally allow the transformations to break connectivity, even though we also develop some connectivity-preserving transformations on the way. Our main result is a universal transformation of $O(n \log n)$ worst-case running time that is permitted to break connectivity. Distributed transformations and connectivity-preserving universal transformations are left as interesting future research directions.

### 1.1 Our Approach

In [MSS19], it was proved that if the devices (called *nodes* from now on) are equipped only with a rotation mechanism, then the decision problem of transforming a connected shape $A$ into a connected shape $B$ is in $\mathbf{P}$, and a constructive characterisation of the (rich) class of pairs of shapes that are transformable to each other was given. In the case of combined availability of rotation and sliding, universality has been shown [DP04, MSS19], that is, any pair of connected shapes are transformable into each other. Still, in these and related models, where in any time step at most one node can move a single position in its local neighbourhood, it can be proved (see, for instance, [MSS19]) that there will be pairs of shapes that require $\Omega(n^2)$ steps to be transformed into each other. This follows directly from the inherent "distance" between the two shapes and the fact that this distance can be reduced by only a constant in every time step. An immediate question is then *"How can we come up with more efficient transformations?"*

Two main alternatives have been explored in the literature in an attempt to answer this question. One is to consider parallel time, meaning that the transformation algorithm can move more than one node (up to a linear number of nodes if possible) in a single time step, such as transformations based on pipelining [DSY04b, MSS19, RCN14]. The other approach is to consider more powerful actuation mechanisms, that have the potential to reduce the inherent distance faster than a constant per sequential time-step. Prominent examples in the literature are the linear-strength models of Aloupis *et al.* [ABD$^+$13, ACD$^+$08], in which nodes are equipped with extend/contract arms and of Woods *et al.* [WCG$^+$13], in which a whole line of nodes can rotate around a single node (acting as a linear-strength rotating arm). The present paper follows this approach, by introducing and investigating a linear-strength model in which a node can push a line of consecutive nodes one position (towards an empty cell) in a single time-step.

In terms of transformability, our model can easily simulate the combined rotation and sliding mechanisms of [DP04, MSS19] by restricting movements to lines of length 1 (i.e., individual nodes). It follows that this model is also capable of universal transformations, with a time complexity at most twice the worst-case of those models, i.e., again $O(n^2)$. Naturally, our focus is set on exploring ways to exploit the parallelism inherent in moving lines of larger length in order to speed-up transformations and, if possible, to come up with a more efficient in the worst case universal transformation. Further, as reversibility of movements is still valid for any line movement in our model, we adopt the approach of transforming any given shape $A$ into a spanning line $L$ (vertical or horizontal). This is convenient, because if one shows that any shape $A$ can transform fast into a line $L$, then any pair of shapes $A$ and $B$ can then be transformed fast to each other by first transforming fast $A$ into $L$ and then $L$ into $B$ by reversing the fast transformation of $B$ into $L$.

We start this investigation by identifying the diagonal shape $D$ (which is considered connected in our model and is very similar to the staircase worst-case shape of [MSS19]) as a potential worst-case initial shape to be transformed

into a line $L$. This intuition is supported by the $O(n^2)$ individual node distance between the two shapes and by the initial unavailability of long line movements: the transformation may move long lines whenever available, but has to pay first a number of movements of small lines in order to construct longer lines. In this benchmark (special) case, the trivial lower and upper bounds $\Omega(n)$ and $O(n^2)$, respectively, hold.

First, we prove that by partitioning the diagonal into $\sqrt{n}$ diagonal segments of length $\sqrt{n}$ each, we can first transform each segment in time quadratic in its length into a straight line segment, then push all segments down to a "collection row" $y_0$ in time $O(n\sqrt{n})$ and finally re-orient all line segments to form a horizontal line in $y_0$, paying a linear additive factor. Thus, this transformation takes total time $O(n\sqrt{n})$, which constitutes our first improvement compared to the $\Omega(n^2)$ lower bound of [MSS19]. We then take this algorithmic idea one step further, by developing two transformations building upon it, that can achieve the same time-bound while *preserving connectivity* throughout their course: one is based on *folding* segments and the other on *extending* them.

As the $O(\sqrt{n})$ length of uniform partitioning into segments is optimal for the above type of transformation, we turn our attention into different approaches, aiming at further reducing the running time of transformations. Allowing once more to break connectivity, we develop an alternative transformation based on *successive doubling*. The partitioning is again uniform for individual "phases", but different phases have different partitioning length. The transformation starts from a minimal partitioning into $n/2$ lines of length 2, then matches them to the closest neighbours via shortest paths to obtain a partitioning into $n/4$ lines of length 4, and, continuing in the same way for $\log n$ phases, it maintains the invariant of having $n/2^i$ individual lines in each phase $i$, for $1 \leq i \leq \log n$. By proving that the cost of pairwise merging through shortest paths in each phase is linear in $n$, we obtain that this approach transforms the diagonal into a line in time $O(n \log n)$, thus yielding a substantial improvement. Observe that the problem of transforming the diagonal into a line seems to involve solving the same problem into smaller diagonal segments (in order to transform those into corresponding line segments). Then, one may naturally wonder whether a recursive approach could be applied in order to further reduce the running time. We provide a negative answer to this, for the special case of uniform recursion and at the same time obtain an alternative $O(n \log n)$ transformation for the diagonal-to-line problem.

Our final aim is to generalise the ideas developed for the above benchmark case in order to come up with *equally efficient universal transformations*. We successfully generalise both the $O(n\sqrt{n})$ and the $O(n \log n)$ approaches, obtaining universal transformations of worst-case running times $O(n\sqrt{n})$ and $O(n \log n)$, respectively. We achieve this by enclosing the initial shape into a square bounding box and then subdividing the box into square sub-boxes of appropriate dimension. For the $O(n\sqrt{n})$ bound, a single such partitioning into sub-boxes of dimension $\sqrt{n}$ turns out to be sufficient. For the $O(n \log n)$ bound we again employ a successive doubling approach through phases of an increasing dimension

of the sub-boxes, that is, through a new partitioning in each phase. Therefore, our ultimate theorem (followed by a constructive proof, providing the claimed transformation) states that: *"In this model, when connectivity need not necessarily be preserved during the transformation, any pair of connected shapes $A$ and $B$ can be transformed to each other in sequential time $O(n \log n)$"*.

Table 1 summarises the running times of all the transformations developed in this paper.

| Transformation | Problem | Running Time | Lower Bound |
|---|---|---|---|
| *DL-Partitioning* | Diagonal | $O(n\sqrt{n})$ | $\Omega(n)$ |
| *DL-Doubling* | Diagonal | $O(n \log n)$ | $\Omega(n)$ |
| *DL-Recursion* | Diagonal | $O(n \log n)$ | $\Omega(n)$ |
| *DLC-Folding* | Diagonal Connected | $O(n\sqrt{n})$ | $\Omega(n)$ |
| *DLC-Extending* | Diagonal Connected | $O(n\sqrt{n})$ | $\Omega(n)$ |
| *U-Box-Partitioning* | Universal | $O(n\sqrt{n})$ | $\Omega(n)$ |
| *U-Box-Doubling* | Universal | $O(n \log n)$ | $\Omega(n)$ |

Table 1: A summary of our transformations and their corresponding worst-case running times (the trivial lower bound is in all cases $\Omega(n)$). The Diagonal, Diagonal Connected, and Universal problems correspond to the DiagonalToLine, DiagonalToLineConnected, and UniversalTransformation problems, respectively (being formally defined in Section 2).

Section 2 brings together all definitions and basic facts that are used throughout the paper. In Section 3, we study the problem of transforming a diagonal shape into a line, without and with connectivity preservation. Section 4 presents our universal transformations. In Section 5 we conclude and discuss further research directions that are opened by our work.

## 2 Preliminaries and Definitions

The transformations considered here run on a two-dimensional square grid. Each cell of the grid possesses a unique location addressed by non-negative coordinates $(x, y)$, where $x$ denotes columns and $y$ indicates rows. A *shape* $S$ is a set of $n$ *nodes* on the grid, where each individual node $u \in S$ occupies a single cell $cell(u) = (x_u, y_u)$, therefore we may also refer to a node by the coordinates of the cell that it occupies at a given time. Two distinct nodes $(x_1, y_1)$, $(x_2, y_2)$ are *neighbours* (or *adjacent*) iff $x_2 - 1 \leq x_1 \leq x_2 + 1$ and $y_2 - 1 \leq y_1 \leq y_2 + 1$ (i.e., their cells are adjacent vertically, horizontally or diagonally). A shape $S$ is *connected* iff the graph defined by $S$ and the above neighbouring relation on $S$ is connected. Throughout, $n$ denotes the number of nodes in a shape under consideration.

A line, $L \subseteq S$, is defined by one or more consecutive nodes in a column or row. That is, $L = (x_0, y_0), (x_1, y_1), \ldots, (x_k, y_k)$, for $0 \leq k \leq n$, $k \in \mathbb{Z}$, is a line iff $x_0 = x_1 = \cdots = x_k$ and $|y_k - y_0| = k$, or $y_0 = y_1 = \cdots = y_k$ and $|x_k - x_0| = k$. A *line move*, is an operation by which all nodes of a line $L$ move together in a single step, towards an empty *cell* adjacent to one of $L$'s endpoints. A line move may also be referred to as *step* (or *move* or *movement*) and time is discrete and measured in number of steps throughout. A move in this model is equivalent to choosing a node $u$ and a direction $d \in \{up, down, left, right\}$ and moving $u$ one position in direction $d$. This will additionally push by one position the whole line $L$ of nodes in direction $d$, $L$ (possibly empty) starting from a neighbour of $u$ in $d$ and ending at the first empty cell.

More formally and in slightly different terms: A line $L = (x_1, y), (x_2, y), \ldots, (x_k, y)$ of length $k$, where $1 \leq k \leq n$, can push all $k$ nodes rightwards in a single step to positions $(x_2, y), (x_3, y), \ldots, (x_{k+1}, y)$ iff there exists an empty cell to the right of $L$ at $(x_{k+1}, y)$. The "down", "left", and "up" movements are defined symmetrically, by rotating the whole system 90°, 180°, and 270° clockwise, respectively.

As already mentioned, we know that there are related settings in which 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 while preserving the connectivity throughout the course of the transformation.[1] This, for example, has been proved for the case in which the available movements to the nodes are rotation and sliding [DP04, MSS19]. It can be shown that the model of [DP04, MSS19] is a special case of our model, implying all transformations established there (with their running time at most doubled, including universal transformations, are also valid transformations in the present model).

**Lemma 1.** *The minimum number of line moves by which a line of length $k$, $1 \leq k \leq n$, can completely change its orientation*[2]*, is $2k - 2$.*

A property that typically facilitates the development of universal transformations, is reversibility of movements. To this end, we next show that line movements are reversible.

**Lemma 2 (Reversibility).** *Let $(S_I, S_F)$ be a pair of connected shapes of the same number of nodes $n$. If $S_I \to S_F$ ("$\to$" denoting "can be transformed to via a sequence of line movements") then $S_F \to S_I$.*

**Definition 1 (Nice Shape).** *A connected shape $S \in NICE$ if there exists a central line $L_C \subseteq S$, such that every node $u \in S \setminus L_C$ is connected to $L_C$ via a line perpendicular to $L_C$.*

[1] In this paper, whenever transforming into a target shape $B$, we allow any placement of $B$ on the grid, i.e., any shape $B'$ obtained from $B$ through a sequence of rotations and translations.

[2] From vertical to horizontal and vice versa.

**Proposition 1.** *Let $S_{Nice}$ be a nice shape and $S_L$ a straight line, both of the same order $n$. Then $S_{Nice} \to S_L$ (and $S_L \to S_{Nice}$) in $O(n)$ steps.*

We now formally define the problems to be considered in this paper.
DiagonalToLine**.** Given an initial connected diagonal line $S_D$ and a target vertical or horizontal connected spanning line $S_L$ of the same order, transform $S_D$ into $S_L$, without necessarily preserving the connectivity during the transformation.
DiagonalToLineConnected**.** Restricted version of DiagonalToLine in which connectivity must be preserved during the transformation.
UniversalTransformation**.** Give a general transformation, such that, for all pairs of shapes $(S_I, S_F)$ of the same order, where $S_I$ is the initial shape and $S_F$ the target shape, it will transform $S_I$ into $S_F$, without necessarily preserving connectivity during its course.

## 3 Transforming the Diagonal into a Line

We begin our study from the case in which the initial shape is a diagonal line $S_D$ of order $n$. Our goal throughout the section is to transform $S_D$ into a spanning line $S_L$, i.e., solve the DiagonalToLine and/or DiagonalToLineConnected problems. We do this, because these problems seem to capture the worst-case complexity of transformations in this model.

### 3.1 An $O(n\sqrt{n})$-time Transformation

We start from DiagonalToLine (i.e., no requirement to preserve connectivity). Our strategy (called *DL-Partitioning*) is as follows. We partition the diagonal into equal segments, as in Figure 1 (a). Then in each segment, we perform a trivial (inefficient, but enough for our purposes) line formation by moving each node independently to the leftmost column in that segment(Figure 1 (b)), which transforms all segments into lines (Figure 1 (c)). Then, we transfer each line segment all the way down to the bottommost row of the diagonal $S_D$, see Figure 1 (d). Finally, we change the orientation of all line segments to form the target spanning line (Figure 1 (e)). More formally, let $S_D$ be a diagonal, occupying $(x, y), (x+1, y+1), \ldots, (x+n-1, y+n-1)$, such that $x$ and $y$ are the leftmost column and the bottommost row of $S_D$, respectively. $S_D$ is divided into $\lceil\sqrt{n}\rceil$ segments, $l_1, l_2, \ldots, l_{\lceil\sqrt{n}\rceil}$, each of length $\lfloor\sqrt{n}\rfloor$, apart possibly from a single smaller one. Figure 1 (a) illustrates the case of integer $\sqrt{n}$ and in what follows, w.l.o.g., we present the case of integer $\sqrt{n}$ for simplicity. This strategy consists of three phases:
**Phase 1**: Transforms each diagonal segment $l_1, l_2, \ldots, l_{\sqrt{n}}$ into a line segment. Notice that segment $l_k$, $1 \leq k \leq \sqrt{n}$, contains $\sqrt{n}$ nodes occupying positions $(x+h_k, y+h_k), (x+h_k+1, y+h_k+1), \ldots, (x+h_k+\sqrt{n}-1, y+h_k+\sqrt{n}-1)$, for $h_k = n - k\sqrt{n}$; see Figure 1 (b). Each of these nodes moves independently to the leftmost column of $l_k$, namely column $x+h_k$, and the new positions of the

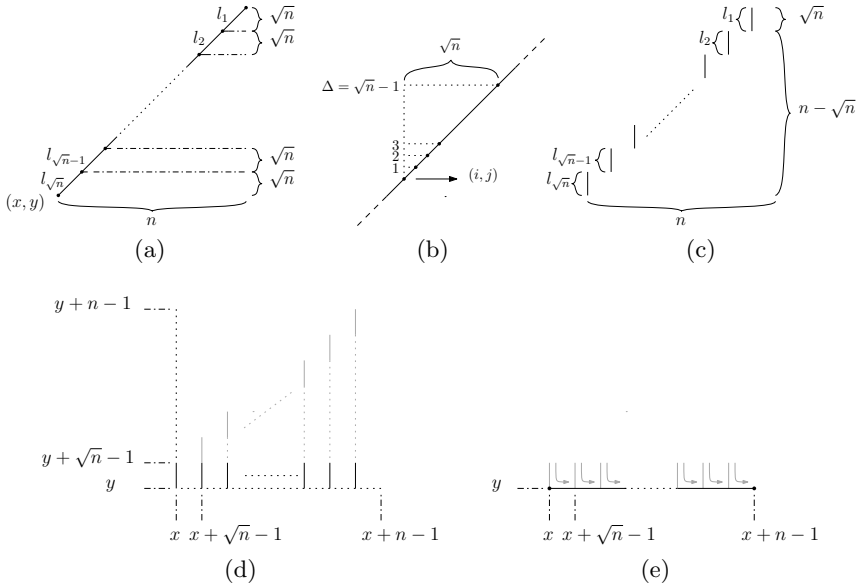


Fig. 1: (a) Dividing the diagonal into $\sqrt{n}$ segments of length $\sqrt{n}$ each (integer $\sqrt{n}$ case). (b) A closer view of a single segment, where $1, 2, 3, \ldots, \sqrt{n}-1$ are the required distances for the nodes to form a line segment at the leftmost column (of the segment). (c) Each line segment is transformed into a line and transferred towards the bottommost row of the shape, ending up as in (d). (e) All line segments are turned into the bottommost row to form the target spanning line.

nodes become $(x+h_k, y+h_k), (x+h_k, y+h_k+1), \ldots, (x+h_k, y+h_k+\sqrt{n}-1)$. Due to symmetry, any segment follows the same procedure of gathering at its bottommost row. By the end of Phase 1, $\sqrt{n}$ vertical line segments have been created (Figure 1 (c)).

**Phase 2**: Transfers all $\sqrt{n}$ line segments from Phase 1 down to the bottommost row $y$ of the diagonal $S_D$. Observe that line segment $l_k$ has to move distance $h_k$ (see Figure 1 (d)).

**Phase 3**: Turns all $\sqrt{n}$ line segments into the bottommost row $y$ (Figure 1 (e)). In particular, line $l_k$ will be occupying positions $(x+h_k, y), (x+h_k+1, y), \ldots, (x+h_k+\sqrt{n}-1, y)$.

**Theorem 1.** *Given an initial diagonal of $n$ nodes,* DL-Partitioning *solves the* DiagonalToLine *problem in $O(n\sqrt{n})$ steps.*

Going one step further, we provide two $O(n\sqrt{n})$-time transformations, *DLC-Folding* and *DLC-Extending*, that additionally preserve connectivity of the shape throughout the transformation.[3]

[3] Due to space restrictions, these can be found in the full version of the paper.

**Theorem 2.** *Given an initial connected diagonal of $n$ nodes,* DLC-Folding *and* DLC-Extending *solve the* DiagonalToLineConnected *problem in $O(n\sqrt{n})$ steps.*

### 3.2 An $O(n \log n)$-time Transformation

We now investigate another approach (called *DL-Doubling*) for DiagonalToLine (i.e., without necessarily preserving connectivity). The main idea is as follows. The initial configuration can be viewed as $n$ lines of length 1. We start (in *phases*) to successively double the length of lines (while halving their number) by matching them in pairs through shortest paths, until a single spanning line remains. Let the lines existing in each phase be labelled $1, 2, 3, \ldots$ from top-right to bottom-left. In each phase, we shall distinguish two types of lines, *free* and *stationary*, which correspond to the odd $(1, 3, 5, \ldots)$ and even $(2, 4, 6, \ldots)$ lines from top-right to bottom-left, respectively. In any phase, only the *free* lines move, while the *stationary* stay still. In particular, in phase $i$, every free line $j$ moves via a shortest path to merge with the next (top-right to bottom-left) stationary line $j + 1$. This operation merges two lines of length $k$ into a new line of length $2k$ residing at the column of the stationary line. In general, at the beginning of every phase $i$, $1 \leq i \leq \log n$, there are $n/2^{i-1}$ lines of length $2^{i-1}$ each. These are interchangeably free and stationary, starting from a free top-right one, and at distance $2^{i-1}$ from each other. The minimum number of steps by which any free line of length $k_i$, $1 \leq k_i \leq n/2$ can be merged with the stationary next to it is roughly at most $4k_i = 4 \cdot 2^i$ (by two applications of turning of Lemma 1). By the end of phase $i$ (as well as the beginning of phase $i + 1$), there will be $n/2^i$ lines of length $2^i$ each, at distances $2^i$ from each other. The total cost for phase $i$ is obtained then by observing that each of $n/2^i$ free lines is paying at most $4 \cdot 2^i$ to merge with the next stationary. Thus, the transformation performs a linear number of steps in each of the $\log n$ phases. See Figure 2 for an illustration.

**Lemma 3.** *By the end of phase $i$, for all $1 \leq i \leq \log n$,* DL-Doubling *has created $n/2^i$ lines, each of length $2^i$, by performing $O(n)$ steps in that phase.*

**Theorem 3.** DL-Doubling *transforms any diagonal $S_D$ of order $n$ into a line $S_L$ in $O(n \log n)$ steps.*

An interesting observation for DiagonalToLine (i.e., without necessarily preserving connectivity), is that the problem is essentially self-reducible. This means that any transformation for the problem can be applied to smaller parts of the diagonal, resulting in small lines, and then trying to merge those lines into a single spanning line. An immediate question is then whether such recursive transformations can improve upon the $O(n \log n)$ best upper bound established so far. The extreme application of this idea is to employ a full uniform recursion (call it *DL-Recursion*), where $S_D$ is first partitioned into two diagonals of length $n/2$ each, and each of them is being transformed into a line of length $n/2$, by recursively applying to them the same halving procedure. Finally, the top-right half has to pay a total of at most $4(n/2) = 2n$ to merge with the

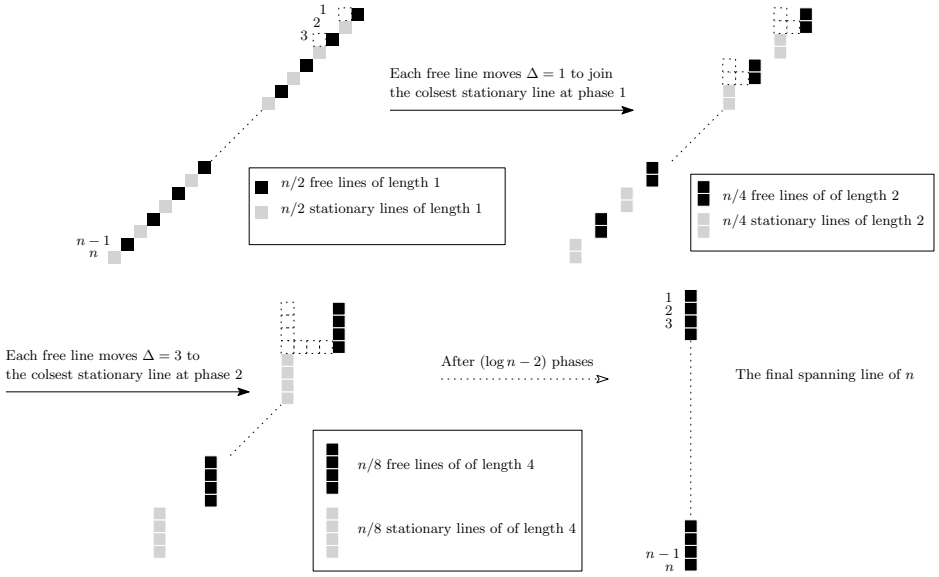


Fig. 2: The process of the $O(n \log n)$-time *DL-Doubling*. Nodes reside inside the black and grey cells.

bottom-left half and form a single spanning line (and the same is being recursively performed by smaller lines). By analysing the running time of such a uniform recursion, we obtain that it is still $O(n \log n)$, partially suggesting that recursive transformations might not be enough to improve upon $O(n \log n)$ (also possibly because of an $\Omega(n \log n)$ matching lower bound, which is left as an open question). If we denote by $T_k$ the total time needed to split and merge lines of length $k$, then the recursion starts from 1 line incurring $T_n$ and ends up with $n$ lines incurring $T_1$. In particular, we analyse the recurrence relation: $T_n = 2T_{n/2} + 2n = 2(2T_{n/4} + n) + 2n = 4T_{n/4} + 4n = 4(2T_{n/8} + n/2) + 4n = 8T_{n/8} + 6n = \cdots = 2^i T_{n/2^i} + 2i \cdot n = \cdots = 2^{\log n} T_{n/2^{\log n}} + 2(\log n)n = n \cdot T_1 + 2n \log n = n + 2n \log n = O(n \log n)$, because $T_1 = 1$.

**Theorem 4.** DL-Recursion *transforms any diagonal $S_D$ of order $n$ into a line $S_L$ of the same order in $O(n \log n)$ steps.*

## 4 Universal Transformations

### 4.1 An $O(n\sqrt{n})$-time Universal Transformation

In this section, we develop a universal transformation, called *U-Box-Partitioning*, which exploits line movements in order to transform *any* initial connected shape $S_I$ into *any* target shape $S_F$ of the same order $n$, in $O(n\sqrt{n})$ steps. Due to reversibility (Lemma 2), it is sufficient to show that any initial connected shape

$S_I$ can be transformed into a spanning line (implying then that any pair of shapes can be transformed to each other via the line and by reversing one of the two transformations). We maintain our focus on transformations that are allowed to break connectivity during their course. Observe that any initial connected shape $S_I$ of order $n$ can be enclosed in an appropriately positioned $n \times n$ square (called a *box*). Our universal transformation is divided into three phases:

**Phase A**: Partition the $n \times n$ box into $\sqrt{n} \times \sqrt{n}$ sub-boxes ($n$ in total in order to cover the whole $n \times n$ box). For each sub-box move all nodes in it down towards the bottommost row of that sub-box as follows. Start filling in the bottommost row from left to right, then if there is no more space continue to the next row from left to right and so on until all nodes in the sub-box have been exhausted (resulting in zero or more complete rows and at most one incomplete row). Moving down is done via shortest paths (where in the worst case a node has to move distance $2\sqrt{n}$); see Figure 3.

**Phase B**: Choose one of the four length-$n$ boundaries of the $n \times n$ box, say w.l.o.g. the left boundary. This is where the spanning line will be formed. Then, transfer every line via a shortest path to that boundary (incurring a maximum distance of $n - \sqrt{n}$ per line).

**Phase C**: Turn all lines (possibly consisting of more than one line on top of each other), by a procedure similar to that of Figure 1 (e), to end up with a spanning line of $n$ nodes on the left boundary.

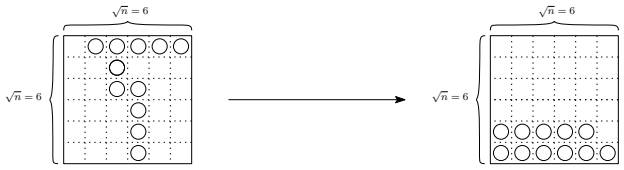


Fig. 3: An example of moving all nodes in a $\sqrt{n} \times \sqrt{n}$ sub-box to fill in the bottommost rows of the sub-box (Phase A).

**Lemma 4.** *A connected shape $S_I$ of order $n$, occupies $O(\sqrt{n})$ sub-boxes.*

*Proof.* It follows directly from Corollary 1, which states that for a given connected shape $S_I$ of $n$ nodes enclosed by a square box of size $n \times n$ and any uniform partitioning of that box into sub-boxes of dimension $d$, then, it holds that $S_I$ can occupy at most $O(\frac{n}{d})$ sub-boxes. Here, *U-Box-Partitioning* is dividing the $n \times n$ square box into $\sqrt{n} \times \sqrt{n}$ sub-boxes of dimension $d = \sqrt{n}$, therefore, $S_I$ can occupy at most $\frac{n}{\sqrt{n}} = O(\sqrt{n})$ sub-boxes.

□

**Lemma 5.** *Starting from any connected shape $S_I$ of order $n$, Phases A and B complete in $O(n\sqrt{n})$ steps each.*

**Lemma 6.** *Consider any length-$n$ boundary and $n$ nodes forming $k$ lines, where $1 \leq k \leq n$, that are perpendicular to that boundary. Then, by line movements, the $k$ lines require at most $O(n)$ steps to form a line of length $n$ on that boundary. This implies that Phase C is completed in $O(n)$ steps.*

*Proof.* See Figure 4. Observe that the $k$ lines of $n$ nodes are connected perpendicularly to the length-$n$ boundary via $k$ nodes, where $1 \leq k \leq n$. It means that there are $n - k$ nodes still waiting to be pushed into that boundary. According to Lemma 1, each of the $n - k$ nodes requires 2 steps to occupy the border, with a total of $2(n - k)$ steps for all $n - k$ nodes to completely fill up the boundary of length $n$. Following that, *U-Box-Partitioning* pushes all $k$ lines in a total $t$ of at most,

$$\begin{aligned} t &= 2(n-k) = 2n - 2k \\ &= O(n). \end{aligned}$$

□

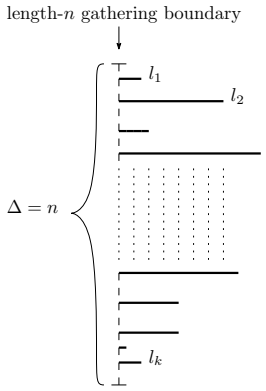


Fig. 4: The dashed line is a length-$n$ gathering boundary of the $n \times n$ box, which is connected perpendicularly to $k$ lines of $n$ nodes.

**Lemma 7.** U-Box-Partitioning *transforms any connected shape $S_I$ into a straight line $S_L$ of the same order $n$, in $O(n\sqrt{n})$ steps.*

Putting Lemma 7 and reversibility (Lemma 2) together gives:

**Theorem 5.** *For any pair of connected shapes $S_I$ and $S_F$ of the same order $n$,* U-Box-Partitioning *can be used to transform $S_I$ into $S_F$ (and $S_F$ into $S_I$) in $O(n\sqrt{n})$ steps.*

### 4.2 An $O(n \log n)$-time Universal Transformation

We now present an alternative universal transformation, called *U-Box-Doubling*, that transforms any pair of connected shapes, of the same order, to each other in $O(n \log n)$ steps. Given a connected shape $S_I$ of order $n$, do the following. *Enclose* $S_I$ into an arbitrary $n \times n$ *box* as in *U-Box-Partitioning* (Section 4.1). For simplicity, we assume that $n$ is a power of 2, but this assumption can be dropped. Proceed in $\log n$ phases as follows: In every phase $i$, where $1 \leq i \leq \log n$, partition the $n \times n$ box into $2^i \times 2^i$ sub-boxes, disjoint and completely covering the $n \times n$ box. Assume that from any phase $i-1$, any $2^{i-1} \times 2^{i-1}$ sub-box is either empty or has its $k$, where $0 \leq k \leq 2^{i-1}$, bottommost rows completely filled in with nodes, possibly followed by a single incomplete row on top of them containing $l$, where $1 \leq l < 2^{i-1}$, consecutive nodes that are left aligned on that row. This case holds trivially for phase 1 and inductively for every phase. That is, in odd phases, we assume that nodes fill in the leftmost columns of boxes in a symmetric way. Every $2^i \times 2^i$ sub-box (of phase $i$) consists of four $2^{i-1} \times 2^{i-1}$ sub-boxes from phase $i-1$, each of which is either empty or occupied as described above.

Consider the case where $i$ is odd, thus, the nodes in the $2^{i-1} \times 2^{i-1}$ sub-boxes are bottom aligned. For every $2^i \times 2^i$ sub-box, move each line from the previous phase that resides in the sub-box to the left as many steps as required until that row contains a single line of consecutive nodes, starting from the left boundary of the sub-box, as shown in Figure 5 (a). With a linear procedure similar to that of Lemma 6 (and of *nice shapes*), start filling in the columns of the $2^i \times 2^i$ sub-box from the leftmost column and continuing to the right. If an incomplete column remains, push the nodes in it to the bottom of that column; see Figure 5 (b) for an example. The case of even $i$ is symmetric, the only difference being that the arrangement guarantee from $i-1$ is left alignment on the columns of the $2^{i-1} \times 2^{i-1}$ sub-boxes and the result will be bottom alignment on the rows of the $2^i \times 2^i$ sub-boxes of the current phase. This completes the description of the transformation. We first prove *correctness*:


Fig. 5: (a) Pushing left in each $2^i \times 2^i$ sub-box. (b) Cleaning the orientation by aligning (filling) the leftmost columns.

**Lemma 8.** *Starting from any connected shape $S_I$ of order $n$,* U-Box-Doubling *forms by the end of phase* $\log n$ *a line of length $n$.*

*Proof.* In phase $\log n$, the procedure partitions into a single box, which is the whole original $n \times n$ box. Independently of whether gathering will be on the leftmost column or on the bottommost row of the box, as all $n$ nodes are contained in it, the outcome will be a single line of length $n$, vertical or horizontal, respectively. □

Now, we shall analyse the running time of *U-Box-Doubling*. To facilitate exposition, we break this down into a number of lemmas.

**Lemma 9.** *In every phase $i$, the "super-shape" formed by the occupied $2^i \times 2^i$ sub-boxes is connected.*

*Proof.* By induction on the phase number $i$ (starting from $S_I$ connected) and the observation that a sub-box is occupied iff any of its own sub-boxes (of any size) had ever been occupied, because nodes are not transferred between $2^i \times 2^i$ sub-boxes before phase $i+1$. □

**Lemma 10.** *Given that* U-Box-Doubling *starts from a connected shape $S_I$ of order $n$, the number of occupied sub-boxes in any phase $i$ is $O(\frac{n}{2^i})$.*

*Proof.* First, observe that a $2^i \times 2^i$ sub-box of phase $i$ is occupied in that phase iff $S_I$ was originally going through that box. This follows from the fact that nodes are not transferred by this transformation between $2^i \times 2^i$ sub-boxes before phase $i+1$. Therefore, the $2^i \times 2^i$ sub-boxes occupied in (any) phase $i$ are exactly the $2^i \times 2^i$ sub-boxes that the original shape $S_I$ would have occupied, thus, it is sufficient to upper bound the number of $2^i \times 2^i$ sub-boxes that a connected shape of order $n$ can occupy. Or equivalently, we shall lower bound the number $N_k$ of nodes needed to occupy $k$ sub-boxes.

In order to simplify the argument, whenever $S_I$ occupies another unoccupied sub-box, we will award it a constant number of additional occupations for free and only calculate the additional distance (in nodes) that the shape has to cover in order to reach another unoccupied sub-box. In particular, pick any node of $S_I$ and consider as freely occupied that sub-box and the 8 sub-boxes surrounding it. Giving sub-boxes for free can only help the shape, therefore, any lower bound established including the free sub-boxes will also hold for shapes that do not have them (thus, for the original problem). Given that free boxes are surrounding the current node, in order for $S_I$ to occupy another sub-box, at least one surrounding $2^i \times 2^i$ sub-box must be exited. This requires covering a distance of at least $2^i$, through a connected path of nodes. Once this happens, $S_I$ has just crossed the boundary between an occupied sub-box and an unoccupied sub-box. Then, by giving it for free at most 5 more unoccupied sub-boxes, $S_I$ has to pay another $2^i$ nodes to occupy another unoccupied sub-box. We then continue applying this 5-for-free strategy until all $n$ nodes have been used.

To sum up, the shape has been given 8 sub-boxes for free, and then for every sub-box covered it has to pay $2^i$ and gets 5 sub-boxes. Thus, to occupy $k = 8 + l \cdot 5$ sub-boxes, at least $l \cdot 2^i$ nodes are needed, that is, $N_k \geq l \cdot 2^i = \frac{k-8}{5} \cdot 2^i$. But shape $S_I$ has order $n$, which means that the number of nodes available is upper

bounded by $n$, i.e., $N_k \leq n$, which gives $\frac{k-8}{5} \cdot 2^i \leq N_k \leq n \Rightarrow \frac{k-8}{5} \cdot 2^i \leq n \Rightarrow \frac{k-8}{5} \leq \frac{n}{2^i} \Rightarrow k \leq 5\left(\frac{n}{2^i}\right) + 8$. We conclude that the number of $2^i \times 2^i$ sub-boxes that can be occupied by a connected shape $S_I$, and, thus, also the number of $2^i \times 2^i$ sub-boxes that are occupied by the transformation in phase $i$, is at most $5(\frac{n}{2^i}) + 8 = O(\frac{n}{2^i})$. □

As a corollary of this, we obtain:

**Corollary 1.** *Given a uniform partitioning of $n \times n$ square box containing a connected shape $S_I$ of order $n$ into $d \times d$ sub-boxes, it holds that $S_I$ can occupy at most $O(\frac{n}{d})$ sub-boxes.*

By using Lemma 10, we can then show that:

**Lemma 11.** *Starting from any connected shape of $n$ nodes,* U-Box-Doubling *performs $O(n \log n)$ steps during its course.*

*Proof.* We prove this by showing that in every phase $i$, $1 \leq i \leq \log n$, the transformation performs at most a linear number of steps. We partition the occupied $2^i \times 2^i$ sub-boxes into two disjoint sets, $B_1$ and $B_0$, where sub-boxes in $B_1$ have at least 1 *complete line* (from the previous phase), i.e., a line of length $2^{i-1}$, and sub-boxes in $B_0$ have 1 to 4 *incomplete lines*, i.e., lines of length between 1 and $2^{i-1} - 1$. For $B_1$, we have that $|B_1| \leq \frac{n}{2^{i-1}}$. Moreover, for every complete line, we pay at most $2^{i-1}$ to transfer it left or down, depending on the parity of $i$. As there are at most $\frac{n}{2^{i-1}}$ such complete lines in phase $i$, the total cost for this is at most $2^i \cdot (\frac{n}{2^{i-1}}) = n$.

Each sub-box in $B_1$ may also have at most 4 incomplete lines from the previous phase, where at most two of them may have to pay a maximum of $2^{i-1}$ to be transferred left or down, depending on the parity of $i$ (as the other two are already aligned). As there are at most $\frac{n}{2^{i-1}}$ sub-boxes in $B_1$, the total cost for this is at most $2 \cdot 2^{i-1} \cdot \left(\frac{n}{2^{i-1}}\right) = 2n$.

Therefore, the total cost for pushing all lines towards the required border in $B_1$ sub-boxes is at most $n + 2n = 3n$. For $B_0$, we have (by Lemma 10) that the total number of occupied sub-boxes in phase $i$ is at most $5\left(\frac{n}{2^{i-1}}\right) + 8$, then $|B_0| \leq 5\left(\frac{n}{2^{i-1}}\right) + 8$ (taking into account also the worst case where every occupied sub-box may be of type $B_0$). There is again a maximum of 2 incomplete lines per such sub-box that need to be transferred a distance of at most $2^{i-1}$, therefore, the total cost for this to happen in every $B_0$ sub-box is at most $2 \cdot 2^{i-1}(5 \cdot \frac{n}{2^i} + 8) = 5n + 8 \cdot 2^i \leq 13n$. By paying the above costs, all occupied sub-boxes have their lines aligned horizontally to their left or vertically to their bottom border, and the final task of the transformation for this phase is to apply a linear procedure in order to fill in the left (bottom) border. This procedure costs at most $2k$ for every $k$ nodes aligned as above (Lemma 1), therefore, in total at most $2n$. This completes the operation of the transformation for phase $i$. Putting everything together, we obtain that the total cost $T_i$, in steps, for phase $i$ is $T_i \leq 3n + 13n + 2n = 18n$. As there is a total of $\log n$ phases, we conclude that the total cost $T$ of the transformation is $T \leq 18n \cdot \log n = O(n \log n)$. □

Finally, Lemma 8, Lemma 11, and reversibility (Lemma 2) imply that:

**Theorem 6.** *For any pair of connected shapes $S_I$ and $S_F$ of the same order $n$, transformation* U-Box-Doubling *can be used to transform $S_I$ into $S_F$ (and $S_F$ into $S_I$) in $O(n \log n)$ steps.*

## 5 Conclusions

In this work, we studied a new linear-strength model of line movements. The nodes can now move in parallel by translating a line of any length by one position in a single time-step. This model, having the model of [DP04, MSS19] as a special case, adopts all its transformability results (including universal transformations). Then, our focus naturally turned to investigating if pushing lines can help achieve a substantial gain in performance (compared to the $\Theta(n^2)$ of those models). Even though it can be immediately observed that there are instances in which this is the case (e.g., initial shapes in which there are many long lines, thus, much initial parallelism to be exploited), it was not obvious that this holds also for the worst case. By identifying the diagonal as a potentially worst-case shape (essentially, because in it any parallelism to be exploited does not come for free), we managed to first develop an $O(n\sqrt{n})$-time transformation for transforming the diagonal into a line, then to improve upon this by two transformations that achieve the same bound while preserving connectivity, and finally to provide an $O(n \log n)$-time transformation (that breaks connectivity). Going one step further, we developed two universal transformations that can transform any pair of connected shapes to each other in time $O(n\sqrt{n})$ and $O(n \log n)$, respectively.

There is a number of interesting problems that are opened by this work. The obvious first target (and apparently intriguing) is to answer whether there is an $o(n \log n)$-time transformation (e.g., linear) or whether there is an $\Omega(n \log n)$-time lower bound matching our best transformations. We suspect the latter, but do not have enough evidence to support or prove it. Moreover, we didn't consider parallel time in this paper. If more than one line can move in parallel in a time-step, then are there variants of our transformations (or alternative ones) that further reduce the running time? In other words, are there parallelisable transformations in this model? In particular, it would be interesting to investigate whether the present model permits an $O(\log n)$ parallel time (universal) transformation, i.e., matching the best transformation in the model of Aloupis *et al.* [ACD+08]. It would also be worth studying in more depth the case in which connectivity has to be preserved during the transformations. In the relevant literature, a number of alternative types of grids have been considered, like triangular (e.g., in [DDG+14]) and hexagonal (e.g., in [WWA04]), and it would be interesting to investigate how our results translate there. Finally, an immediate next goal is to attempt to develop distributed versions of the transformations provided here.

## 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.

[ACD+08] G. Aloupis, S. Collette, E. D. Demaine, S. Langerman, V. Sacristán, and S. Wuhrer. Reconfiguration of cube-style modular robots using O(logn) parallel moves. In *International Symposium on Algorithms and Computation*, pages 342–353. Springer, 2008.

[BDF+19] A. T. Becker, E. D. Demaine, S. P. Fekete, J. Lonsford, and R. Morris-Wright. Particle computation: complexity, algorithms, and logic. *Natural Computing*, 18[1]:181–201, 2019.

[BG15] J. Bourgeois and S. C. Goldstein. Distributed intelligent MEMS: progresses and perspective. *IEEE Systems Journal*, 9[3]:1057–1068, 2015.

[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.

[CFPS12] M. Cieliebak, P. Flocchini, G. Prencipe, and N. Santoro. Distributed computing by mobile robots: Gathering. *SIAM Journal on Computing*, 41[4]:829–879, 2012.

[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.

[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. ACM, 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.

[DDL+09] S. M. Douglas, H. Dietz, T. Liedl, B. Högberg, F. Graf, and W. M. Shih. Self-assembly of dna into nanoscale three-dimensional shapes. *Nature*, 459[7245]:414, 2009.

[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.

[DGMRP06] X. Défago, M. Gradinariu, S. Messika, and P. Raipin-Parvédy. Fault-tolerant and self-stabilizing mobile robots gathering. In *International Symposium on Distributed Computing*, pages 46–60. Springer, 2006.

[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.

[DLFP+18] G. A. Di Luna, P. Flocchini, G. Prencipe, N. Santoro, and G. Viglietta. Line recovery by programmable particles. In *Proceedings of the 19th International Conference on Distributed Computing and Networking*, ICDCN '18, pages 4:1–4:10, New York, NY, USA, 2018. ACM.

[DLFS+19] G. A. Di Luna, P. Flocchini, N. Santoro, G. Viglietta, and Y. Yamauchi. Shape formation by programmable particles. *Distributed Computing*, Mar 2019.

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

[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.

[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.

[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.

[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.

[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. *Commun. ACM*, 61[2]:72–81, 2018.

[MSS19] O. Michail, G. Skretas, and P. G. Spirakis. On the transformation capability of feasible mechanisms for programmable matter. *Journal of Computer and System Sciences*, 102:18–39, 2019.

[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.

[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. ACM, 2000.

[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.

[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.

[YS10] M. Yamashita and I. Suzuki. Characterizing geometric patterns formable by oblivious anonymous mobile robots. *Theoretical Computer Science*, 411[26-28]:2433–2453, 2010.

[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.