Brief Announcement: Efficient Distributed Algorithms for Shape Reduction via Reconfigurable Circuits

Nada Almalki
Department of Computer Science, University of Liverpool, UK

Siddharth Gupta
Department of Computer Science & Information Systems, BITS Pilani Goa Campus, India

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

Andreas Padalkin
Paderborn University, Germany

Abstract

In this paper, we study the problem of efficiently reducing geometric shapes into other such shapes in a distributed setting through size-changing operations. We develop distributed algorithms using the *reconfigurable circuit* model to enable fast node-to-node communication. Let n denote the number of nodes and k the number of turning points in the initial shape. We show that the system of nodes can reduce itself from any tree to a single node using only *shrinking* operations in $O(k \log n)$ rounds w.h.p. and any tree to its *incompressible* form in $O(\log n)$ rounds given prior knowledge of the incompressible nodes, or $O(k \log n)$ without it, w.h.p. We also give an algorithm to transform any tree to a topologically equivalent tree in $O(k \log n + \log^2 n)$ rounds w.h.p. using both *shrinking* and *growth* operations. On the negative side, we show that one cannot hope for $o(\log^2 n)$-round transformations for all shapes of $\Theta(\log n)$ turning points.

2012 ACM Subject Classification Theory of computation → Computational geometry; Theory of computation → Distributed computing models

Keywords and phrases growth process, shrinking process, collision avoidance, programmable matter

Digital Object Identifier 10.4230/LIPIcs.SAND.2025.20

Related Version *Full Version*: https://arxiv.org/abs/2503.18663

Funding *Andreas Padalkin*: This author was supported by the DFG Project SCHE 1592/10-1.

1 Introduction

A central problem in reconfigurable robotics and related areas is for a set of agents to transform from an initial shape into a target shape, efficiently and without violating structural and other constraints. A large part of work has focused on transformations that guide the system through a sequence of configuration changes without altering its size over time. Other work in algorithmic self-assembly and distributed computing, has studied passive size-changing dynamics whose accumulation leads to structural changes over time. Some authors have recently begun to study transformations that can actively form or deform a configuration by locally controlling the addition or removal of individual agents. These processes are motivated by biological and physical systems, such as embryonic growth in multicellular organisms, and by dynamical systems, such as cellular and self-reproducing automata. They are also related to recent work on algorithmic reconfiguration of graphs and networks [3, 5, 8, 7].

The *nubot* model of [10] is a decentralized algorithmic framework for autonomous growth and self-assembly at the molecular scale. [2] studied the feasibility and centralized complexity of growing geometric shapes through simple forms of growth operations. These were

conditioned to affect specific parts of the shape (e.g., whole columns) and were defined to ensure in advance that no collision can happen. In [1], this was extended to general growth operations, aiming for exponentially fast, centralized transformations that avoid collisions.

Parallel transformation processes can be exponentially fast and are highly dynamic, making it difficult to avoid collisions [6]. It is therefore important to have fast and reliable communication within the system – especially over long distances. [4] proposed the deployment of *reconfigurable circuits*, which allow to connect subsets of agents. Each agent can send simple signals (beeps) through any connected circuit, which is instantaneously received by all agents connected to the same circuit. Reconfigurable circuits have enabled polylogarithmic-time solutions to problems like shape recognition [4] and spanning tree [9].

In this work, we leverage reconfigurable circuits as a communication model for rapid distributed size-changing transformations. In particular, we study distributed algorithms that must efficiently reduce a given shape into a target shape (sometimes a single node) through a sequence of parallel, collision-free shrinking operations, or by combining shrinking and growth. This also offers a new application domain for reconfigurable circuits, which have primarily been applied to the *amoebot* model and graph problems.

2 Models and Problems

We study a system composed of computational nodes that initially form a connected shape $S = (V, E)$ of size $|V| = n$ on a two-dimensional square grid. Each node $u \in V$ occupies a distinct point (u_x, u_y) of the grid and E is a subset of all pairs of nodes that are orthogonally adjacent on the grid. Nodes operate as anonymous (and randomized) *finite-state automata*, executing actions in *discrete synchronous* rounds. We assume that all nodes agree on a common compass orientation and chirality.

Nodes communicate via *reconfigurable circuits* [4]. In this model, each edge in E is subdivided into a constant number c of *links*, where c is the same for all edges. We assume $c = 2$, which is sufficient and necessary for our results. Each node incident to a link owns one of its endpoints, which are called *pins*. Each node partitions its pins into disjoint sets, called *partition sets*. Let P denote the set of all partition sets of all nodes. Consider the graph $C = (P, L)$, where two partition sets $p_1, p_2 \in P$ are adjacent in L iff there is a link with one pin in p_1 and the other in p_2. In each round, each node can reconfigure its partitioning, and send a *beep* on any subset of its partition sets. At the beginning of the next round, if a beep occurred on a circuit, then all nodes on that circuit will receive a beep but they will gain no information about its origin or the total number of beeps.

In each round, based on their current state and received signals, nodes can execute either a *growth* operation – adding a new node – or a *shrinking* operation – absorbing an adjacent node. We adopt the *connectivity* model of [1], in which edges form only when new nodes are created.[1] One or more growth operations applied in parallel to nodes of a shape S either cause a *collision* (i.e., a self-intersection or structural violation) or yield a new shape S'. Let $T = (V, E)$ be a tree and $u_0 \in V$ its root. A single growth operation applied on a node $u \in V$ toward an adjacent point (x, y), results in either (i) generating a new node u' at point (x, y) and connecting it to u, or (ii) if (x, y) is already occupied by a node v connected to u, generating u' between u and v, connecting it to both u and v, and translating the subtree $T(v)$ by one unit away from u along the axis parallel to uv. A shrinking operation is the reverse: a node u absorbs an adjacent node v. If v has no other neighbors, it is removed.

[1] In the full paper, we also consider a model where edges update dynamically as nodes become adjacent.

Otherwise, u inherits all v's neighbors, each of which is translated one unit toward u. Let S_r denote the shape at round r. A growth process σ is a set of growth operations applied in parallel, transforming S_r into $S_{r'}$, where $r' > r$. A *shrinking process* is defined analogously.[2]

The *relative position* of $v = (v_x, v_y)$ with respect to $u = (u_x, u_y)$ is defined by the signs of $v_x - u_x$ and $v_y - u_y$. Two trees T and T' are *geometrically equivalent* iff they have the same turning points and segments (possibly of different lengths), and *topologically equivalent* iff additionally, their turning points share the same relative positions. A *column* or *row of a shape* S is a column or row of the grid occupied by nodes of S. It is called *compressible* if it contains no turning points, and *incompressible* otherwise. Nodes of incompressible columns or rows are called *incompressible nodes*. A shape S is *incompressible* if all its nodes are incompressible. The incompressible form of S, denoted by $i(S)$, is the incompressible shape that is topologically equivalent to S.

We study the problem of reducing an initial shape $S_I = (V_I, E_I)$ to a target shape $S_F = (V_F, E_F)$, where $|V_F| \leq |V_I|$. We assume that the constructed shapes are equivalent up to translations. In what follows, we restrict attention to S_I and S_F being trees, denoted T_I and T_F, respectively. Throughout, n denotes the number of nodes and k the number of turning points in T_I. Assuming an arbitrary ordering on the turning points of a tree T, let $\ell(s_i)$ denote the length of segment s_i in T. For any pair of geometrically (and, thus, also for topologically) equivalent trees T_I, T_F we assume the same ordering; that is, s_i denotes the same segment with lengths $\ell(s_i^I)$ and $\ell(s_i^F)$ in T_I and T_F, respectively. The input convention assumes that in T_I each segment s_i^I stores a binary representation of $\ell(s_i^I)$ and $\ell(s_i^F)$. We consider the following problem variants. (i) *Single Node Reduction*: the system must reduce an initial tree T_I to $T_F = (\{u_0\}, \emptyset)$ while avoiding collisions. (ii) *Geometrical Reduction*: as in (i), but T_F now being geometrically equivalent to T_I, with $\ell(s_i^F) \leq \ell(s_i^I)$ for all segments s_i. (iii) *Topological Reduction*: as in (ii), but T_F is additionally topologically equivalent to T_I. (iv) *Incompressible Reduction*: a restricted case of (iii), where $T_F = i(T_I)$.

3 Technical Overview

In this section, we present the main results of our work, as shown in Table 1. For *single node reduction*, we assume common chirality to ensure consistent signal direction across circuits. For *incompressible reduction* and *target tree* we additionally assume a unique leader node and a common compass orientation. This allows us to define a direction for each segment (w.l.o.g. $\rightarrow$ and $\downarrow$) and with that an order. All these assumptions can be effectively dropped, as they can be ensured by an additive $O(\log n)$ rounds of preprocessing w.h.p. (see [4]).

Table 1 A summary of the bounds established in this work.

Algorithm	Operation	Running Time
BFS shrinking	Shrinking	$O(k \log n)$
Lower bound for $k = \Theta(\sqrt[3]{n})$	–	$\Omega(k \log k)$
Incompressible tree (known incompressible nodes)	–	$O(\log n)$
Incompressible tree (unknown nodes)	–	$O(k \log n)$
Optimized BFS shrinking	–	$O(\log n + k \log k)$
Target tree	Shrinking and Growth	$O(k \log n + \log^2 n)$

[2] In some cases, a sequence of operations may combine both growth and shrinking operations.

Single Node Reduction. We present the *BFS shrinking* algorithm, which reduces an arbitrary tree T to a single node. This algorithm consists of three subroutines: (i) *segment detection*, (ii) *segment coloring and shrinking*, and (iii) *final segment shrinking*.

The *segment detection* subroutine (i) allows nodes to detect leaves in their segment by establishing two parallel circuits along each segment s_i. Leaves initiate signals along these circuits, enabling segment nodes to determine the direction toward them. This subroutine ensures common orientation across all segments and completes in $O(1)$ rounds. Once segments are detected, the nodes apply the PASC algorithm [9] in the *segment coloring and shrinking* subroutine (ii) to compute their parity based on their distance from the turning point. Then, nodes alternate colors – blue for even parity and green for odd – after which even-parity nodes absorb adjacent odd-parity nodes, progressively halving each segment's length (see Figure 1). Subroutine (ii) completes in $O(\log \ell)$ rounds, where ℓ is the length of the segment. The *final segment shrinking* subroutine (iii) handles the case when T is reduced to a single segment. It is not possible to determine an orientation during (i), which implies that we cannot decide which endpoint should perform the PASC in subroutine (ii). So, we perform leader election among the leaves using the parallel circuits from subroutine (i), achieving a common orientation in $O(\log n)$ rounds w.h.p.

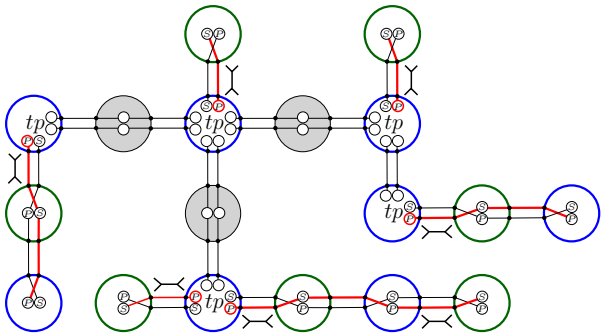

Figure 1 Illustration of PASC applied to compute segment parity relative to turning points.

▶ **Theorem 1.** *After an $O(\log n)$-round preprocessing (w.h.p.), the* BFS shrinking *reduces any tree T to a single node in $O(k \log n)$ rounds.*

A Lower Bound on Geometrical Reduction. We establish a lower bound on the complexity of reducing geometrically equivalent paths using only shrinking operations. Specifically, we show that some path pairs require $\Omega(k \log k)$ rounds, as stated formally below.

▶ **Theorem 2.** *Let $\mathcal{A}$ be an algorithm that reduces a path T_I to a geometrically equivalent path T_F. Then there exists an infinite family of pairs of paths of $k = \Theta(\sqrt[3]{n})$ turning points each, for which $\mathcal{A}$ requires $\Omega(k \log k)$ rounds.*

Incompressible Reduction. In the following, we present the *incompressible tree* algorithm which reduces an initial arbitrary tree T_I to its incompressible form $i(T_I)$. We assume that each node knows whether it is incompressible and, accordingly, whether it belongs to a compressible column or row. By [1], the incompressible shape is obtained by compressing all compressible columns and rows. To shrink compressible columns and rows, we can apply the *segment coloring and shrinking* subroutine.

▶ **Theorem 3.** *Given the incompressible nodes and after an $O(\log n)$-round preprocessing (w.h.p.), the* incompressible tree *algorithm reduces T_I to $i(T_I)$ in $O(\log n)$ rounds.*

An incompressible tree has at most $O(k^2)$ nodes, as each column and row must contain at least one turning point. Hence, running the *incompressible tree* algorithm prior to the *BFS shrinking* algorithm (Theorem 1) yields a runtime of $O(\log n + k \log k)$ rounds w.h.p.

▶ Remark 4. If the incompressible nodes are not known in advance, we can compute all incompressible nodes and with that all compressible segments in $O(k \log n)$ rounds.

Topological Reduction. We present the *target tree* algorithm which reduces an arbitrary tree T_I to another topologically equivalent tree T_F. Let $\ell(s_i^I)$ ($\ell(s_i^F)$) denote the length of segment s_i in the initial (target) tree. We assume that for each i, $\ell(s_i^F) \leq \ell(s_i^I)$. Furthermore, we assume that initially, each segment s_i stores $\ell(s_i^F)$.

A challenge in this case is that we cannot simply apply the *segment coloring and shrinking* subroutine on all segments in parallel. Since doing so does not guarantee that intermediate trees remain topologically equivalent and may lead to collisions. The *target tree* algorithm consists of three phases: *compressible subsegment computation*, *growth*, and *shrinking*. In the *compressible subsegment computation* phase, we compute the lengths of all compressible subsegments of each c_j in T_I and T_F and store them in a segment c_j' in $O(k \log n)$ rounds.

Let $\ell(c_j^I)$ and $\ell(c_j^F)$ denote the length of the compressible subsegment c_j in the initial and target trees, respectively. In the *growth* (*shrinking*) phase, we grow (shrink) all compressible subsegments c_j with $\ell(c_j^F) > \ell(c_j^I)$ ($\ell(c_j^F) < \ell(c_j^I)$). Observe that it does not matter which nodes exactly perform a growth (shrinking) operation in a segment as long as the number of operations is the same. The idea is therefore to first compute a subsegment c_j'' for each c_j (the c_j''s may not be large enough) and then to perform the growth (shrinking) operations of c_j in c_j''. Computing the subsegments requires $O(k \log n)$ rounds while performing the growth (shrinking) operations requires $O(\log^2 n)$ rounds.

▶ **Theorem 5.** *After an $O(\log n)$-round preprocessing (w.h.p.), the* target tree *algorithm reduces T_I to T_F in $O(k \log n + \log^2 n)$ rounds.*

References

1 Nada Almalki, Siddharth Gupta, and Othon Michail. On the exponential growth of geometric shapes. In *ALGOWIN*, volume 15026 of *LNCS*, pages 16–30. Springer, 2024. doi:10.1007/978-3-031-74580-5_2.

2 Nada Almalki and Othon Michail. On geometric shape construction via growth operations. *Theor. Comput. Sci.*, 984:114324, 2024. doi:10.1016/J.TCS.2023.114324.

3 Chen Avin and Stefan Schmid. Toward demand-aware networking: a theory for self-adjusting networks. *Comput. Commun. Rev.*, 48(5):31–40, 2018. doi:10.1145/3310165.3310170.

4 Michael Feldmann, Andreas Padalkin, Christian Scheideler, and Shlomi Dolev. Coordinating amoebots via reconfigurable circuits. *J. Comput. Biol.*, 29(4):317–343, 2022. doi:10.1089/CMB.2021.0363.

5 Thorsten Götte, Kristian Hinnenthal, Christian Scheideler, and Julian Werthmann. Time-optimal construction of overlay networks. *Distributed Comput.*, 36(3):313–347, 2023. doi:10.1007/S00446-023-00442-4.

6 Siddharth Gupta, Marc J. van Kreveld, Othon Michail, and Andreas Padalkin. Collision detection for modular robots - it is easy to cause collisions and hard to avoid them. In *ALGOWIN*, volume 15026 of *LNCS*, pages 76–90. Springer, 2024. doi:10.1007/978-3-031-74580-5_6.

7 George B. Mertzios, Othon Michail, George Skretas, Paul G. Spirakis, and Michail Theofilatos. The complexity of growing a graph. *J. Comput. Syst. Sci.*, 147:103587, 2025. doi:10.1016/J.JCSS.2024.103587.

8 Othon Michail, George Skretas, and Paul G. Spirakis. Distributed computation and reconfiguration in actively dynamic networks. *Distributed Comput.*, 35(2):185–206, 2022. doi:10.1007/S00446-021-00415-5.

9 Andreas Padalkin, Christian Scheideler, and Daniel Warner. The structural power of reconfigurable circuits in the amoebot model. *Nat. Comput.*, 23(4):603–625, 2024. doi:10.1007/S11047-024-09981-6.

10 Damien Woods, Ho-Lin Chen, Scott Goodfriend, Nadine Dabby, Erik Winfree, and Peng Yin. Active self-assembly of algorithmic shapes and patterns in polylogarithmic time. In *Innovations in Theoretical Computer Science, ITCS '13*, pages 353–354. ACM, 2013. doi:10.1145/2422436.2422476.