Brief Announcement: An Effective Geometric Communication Structure for Programmable Matter

Irina Kostitsyna
TU Eindhoven, The Netherlands

Tom Peters
TU Eindhoven, The Netherlands

Bettina Speckmann
TU Eindhoven, The Netherlands

Abstract

The concept of programmable matter envisions a very large number of tiny and simple robot particles forming a smart material that can change its physical properties and shape based on the outcome of computation and movement performed by the individual particles in a concurrent manner. We use geometric insights to develop a new type of shortest path tree for programmable matter systems. Our *feather trees* utilize geometry to allow particles and information to traverse the programmable matter structure via shortest paths even in the presence of multiple overlapping trees.

2012 ACM Subject Classification Computing methodologies → Self-organization

Keywords and phrases Programmable matter, amoebot model, shape reconfiguration

Digital Object Identifier 10.4230/LIPIcs.DISC.2022.47

Related Version *Full Version*: https://arxiv.org/abs/2202.11663

1 Introduction

The concept of programmable matter envisions a very large number of tiny and simple robot particles forming a smart material that can change its physical properties and shape based on the outcome of computation and movement performed by the individual particles in a concurrent manner. We focus on the *amoebot* model, which was introduced in [2] and refined in [1]. This model assumes a very small size of the particles and greatly restricts their computation, communication, and movement capabilities.

In the amoebot model particles occupy nodes of a triangular grid G embedded in the plane. A particle can occupy one (contracted particle) or two (expanded particle) adjacent nodes of the grid. The particles have limited computational power due to constant memory space, no common notion of orientation (disoriented), and no common notion of clockwise or counter-clockwise order (no consensus on chirality). They are identical (no IDs and they all execute the same algorithm), but can locally distinguish between their neighbors using six (for contracted particles) or ten (for expanded particles) *port identifiers*. Ports are labeled in order (either clockwise or counterclockwise) modulo six or ten, respectively. Particles communicate by sending messages to their neighbors using the ports and we assume a particle knows which of its neighbors ports is pointing to itself.

We call the set of particles and their internal states a *particle configuration* $\mathcal{P}$. Let $G_{\mathcal{P}}$ be the subgraph of G induced by the nodes occupied by particles in $\mathcal{P}$. We say that $\mathcal{P}$ is *connected* if there is a path in $G_{\mathcal{P}}$ between any two particles in $\mathcal{P}$. A *hole* in $\mathcal{P}$ is an interior face of $G_{\mathcal{P}}$ with more than three vertices. A particle configuration $\mathcal{P}$ is *simply connected* if it is connected and has no holes.

2 Shortest path trees

Among the previously proposed primitives for amoebot coordination is the *spanning forest primitive* [3] which organizes all particles into trees to facilitate movement while preserving connectivity. However, the spanning forest primitive does not impose any additional structure on the resulting spanning trees. We propose a type of shortest path trees, which we call *feather trees*, in which the path from any particle p to the root r of the feather tree is not longer than any unrestricted path from p to r in $G_{\mathcal{P}}$. Feather trees can be constructed as fast as spanning forests, namely in time linear in the diameter d of $G_{\mathcal{P}}$. They can be used for the same purposes, but have additional geometric properties that support efficient communication and movement of particles even in the presence of multiple overlapping trees.

2.1 Feather trees

We construct a feather tree from a root r in the following way. We distinguish between particles lying on *shafts* (emanating from the root r or other specific nodes) and *branches* (see Fig. 1 (left)). Each particle stores the direction of its parent and whether it is on a shaft or a branch. The root r chooses a maximum independent set of neighbors N_{ind}; the set contains at most three particles and there are at most two ways to choose it. The particles in N_{ind} form the bases of shafts emanating from r. All other neighbors of r form the bases of branches emanating from r. Specifically, let particle p be a neighbor of r across port i. The parent of p is set to $i + 3$ (recall that arithmetic on ports is modulo six), translated to the coordinate system of p. Particle p lies on a shaft if it is in N_{ind}, and on a branch otherwise.

A shaft particle with a parent in direction i propagates the shaft straight to the particle at $i + 3$, and branches into the two directions at ports $i + 2$ and $i + 4$. A branch particle with a parent in direction i propagates the branch straight to the particle at $i + 3$.

We say a *bend* in a path is formed by three consecutive vertices that form a 120° angle. By growing a feather tree according to the rules described so far, we process only particles that are reachable from r by a path with a single bend. We hence need to extend our construction around reflex vertices on the boundary of $\mathcal{P}$ that lie on branches. Specifically, if a branch particle p has a parent at direction i, the direction $i + 1$ (or $i - 1$) does not contain a particle, and the direction $i + 2$ (or $i - 2$) does contain a particle, then p initiates a growth of a new shaft in direction $i + 2$ (or $i - 2$) (see Fig. 1 (right)). We hence have the following lemma:

▶ **Lemma 1.** *Given a simply connected particle configuration $\mathcal{P}$ with diameter d, and a particle $r \in \mathcal{P}$, we can grow a feather tree from r in $O(d)$ rounds.*

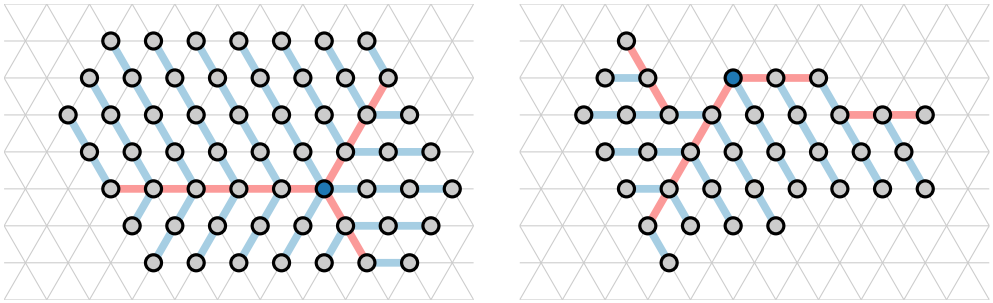

Figure 1 Two feather trees growing from the dark blue root. Shafts are red and branches are blue. Left: every particle is reachable by the initial feathers; Right: additional feathers are necessary.

Every particle is reached by a feather tree exactly once, from one particular direction. Hence, the feather tree is unique, independent of the activation sequence of the particles. In the following we describe how to navigate a set of overlapping feather trees. To do so, we first identify a useful property of shortest paths in feather trees.

We say that a vertex v of $G_{\mathcal{P}}$ is an *inner vertex*, if v and its six neighbors are part of $G_{\mathcal{P}}$. All other vertices are *boundary vertices*. We say that a bend is an *inner bend* if its middle vertex is an inner vertex; otherwise the bend is a *boundary bend*.

▶ **Definition 2** (Feather Path). *A path in $G_{\mathcal{P}}$ is a* feather path *if it does not contain two consecutive inner bends.*

The next lemma follows from the fact that inner bends can occur only on shafts, and any path must alternate visiting shafts and branches.

▶ **Lemma 3.** *Every path from the root to a leaf in a feather tree is a feather path.*

3 Communicating over shortest path trees

Consider a token t traversing a single feather tree F in a network of overlapping feather trees. Due to their limited memory, particles cannot store the identity of F. Despite that, due to Lemma 3, particles can propagate tokens down a feather tree by simply counting the number of inner bends. Thus, when a token is traversing a feather tree F down from the root, it always reaches a leaf of F via a shortest path through $\mathcal{P}$. In particular, it is always a valid choice for p to propagate t straight ahead (if feasible). A left or right 120° turn is a valid choice if it is a boundary bend, or if the last bend the token made was a boundary bend. We cannot control which leaf of F the token t reaches, but it does so without leaving F and hence along a shortest path. We can also broadcast token t to all leaves of F.

Consider now a node ℓ, which is a leaf of multiple feather trees. A token t starting out at node ℓ will always reach the root of one of its feather trees along a shortest path. However, we cannot control which root t reaches. Alternatively, we can broadcast t to all roots of the trees containing node ℓ. As before, the token t navigates by keeping track of the number of inner bends. In particular, if t already made one inner bend since its last boundary bend, then the only valid choice is to continue straight ahead. Otherwise, all three options (straight ahead or a 120° left or right turn) are valid.

References

1 Joshua J. Daymude, Andréa W. Richa, and Christian Scheideler. The Canonical Amoebot Model: Algorithms and Concurrency Control. In *35th International Symposium on Distributed Computing (DISC)*, volume 209 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 20:1–20:19, 2021. doi:10.4230/LIPIcs.DISC.2021.20.

2 Zahra Derakhshandeh, Shlomi Dolev, Robert Gmyr, Andréa W. Richa, Christian Scheideler, and Thim Strothmann. Brief announcement: Amoebot—A New Model for Programmable Matter. In *Proc. 26th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 220–222, 2014. doi:10.1145/2612669.2612712.

3 Zahra Derakhshandeh, Robert Gmyr, Thim Strothmann, Rida Bazzi, Andréa W. Richa, and Christian Scheideler. Leader Election and Shape Formation with Self-organizing Programmable Matter. In *Proc. International Workshop on DNA-Based Computing (DNA)*, LNCS 9211, pages 117–132, 2015. doi:10.1007/978-3-319-21999-8_8.