Polylogarithmic Time Algorithms for Shortest Path Forests in Programmable Matter[*]

Andreas Padalkin[1] and Christian Scheideler[2]

1 **Paderborn University, Germany**
`andreas.padalkin@upb.de`

2 **Paderborn University, Germany**
`scheideler@upb.de`

Abstract

In this paper, we study the computation of shortest paths within the *geometric amoebot model*, a commonly used model for programmable matter. Shortest paths are essential for various tasks and therefore have been heavily investigated in many different contexts. For example, for the geometric amoebot model, Kostitsyna et al. have utilized shortest path trees to transform one amoebot structure into another [DISC, 2023]. We consider the *reconfigurable circuit extension* of the model where the amoebot structure is able to interconnect amoebots by so-called circuits. These circuits permit the instantaneous transmission of simple signals between connected amoebots.

We propose two distributed algorithms for the *shortest path forest problem* where, given a set of k sources and a set of ℓ destinations, the amoebot structure has to compute a forest that connects each destination to its closest source on a shortest path. For hole-free structures, the first algorithm constructs a shortest path tree for a single source within $O(\log \ell)$ rounds, and the second algorithm a shortest path forest for an arbitrary number of sources within $O(\log n \log^2 k)$ rounds. The former algorithm also provides an $O(1)$ rounds solution for the *single pair shortest path problem* (SPSP) and an $O(\log n)$ rounds solution for the *single source shortest path problem* (SSSP) since these problems are special cases of the considered problem.

Related Version *Full Version*: `https://arxiv.org/abs/2402.12123`

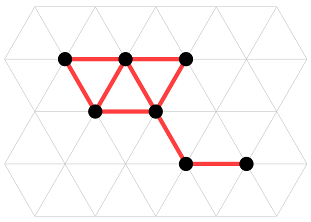
1 Introduction

Programmable matter is matter that has the ability to change its physical properties in a programmable fashion [15]. Many exciting applications have already been envisioned for programmable matter such as self-healing structures and minimal invasive surgery, and shape-changing robots have already been prominent examples of the potentials of programmable matter in many blockbuster movies.

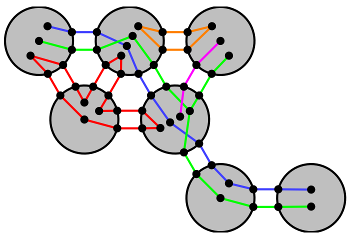
In the *amoebot model*, the matter consists of simple particles (called *amoebots*). In the geometric variant of the model, the amoebots form a connected amoebot structure on the infinite triangular grid, on which they move by *expansions* and *contractions*. However, since information can only travel amoebot by amoebot, many problems come with a natural lower bound of $\Omega(d)$ where d is the diameter of the structure.

For that reason, we consider the *reconfigurable circuit extension* to the amoebot model where the amoebot structure is able to interconnect amoebots by so-called *circuits*. These circuits permit the instantaneous transmission of simple signals between connected amoebots. The extension allows polylogarithmic solutions for various fundamental problems, e.g., leader election [8], and spanning tree construction [12].

[*] This work was supported by the DFG Project SCHE 1592/10-1. The research was initialized at the Dagstuhl Seminar 23091 “Algorithmic Foundations of Programmable Matter”. We thank Shantanu Das, Yuval Emek, Maria Kokkou, Irina Kostitsyna, Tom Peters, and Andrea Richa for helpful discussions.



(a) Amoebot structure.



(b) Reconfigurable circuit extension.

Figure 1 Amoebot model. The left figure shows an amoebot structure. The nodes indicate X. The red edges indicate E_X. The right figure shows an amoebot structure with $c = 2$ external links between adjacent amoebots. The nodes on the boundary are the pins. The nodes within the amoebots indicate the partition sets. An edge between a partition set PS and a pin p implies $p \in PS$. Each color indicates another circuit. The figures were taken from [8].

In this paper, we consider the *shortest path forest problem* where, given a set of sources and a set of destinations, the amoebots have to find a shortest path from each destination to the closest source. Shortest paths are a fundamental problem in both centralized and distributed systems and have also a number of important applications in the amoebot model.

For example, consider shape formation. Many algorithms for the amoebot model utilize a canonical shape, e.g., a line, as an intermediate structure [6, 11]. However, this is rather inefficient if the target structure is already close to the initial structure. For such cases, Kostitsyna et al. proposed an algorithm that utilizes shortest paths to move amoebots through the structure to their target positions [10]. Another application for shortest paths is energy distribution [4, 16]. The amoebots require energy to perform their movements that can be provided by other amoebots, e.g., amoebots located at external energy sources. In order to minimize energy loss, it is more efficient to transfer the energy via a shortest path.

2 Geometric Amoebot Model

The *(geometric) amoebot model* was proposed by Derakhshandeh et al. [5]. The model places a set of n anonymous finite state machines (called *amoebots*) on some graph $G = (V, E)$. Each amoebot occupies one node and every node is occupied by at most one amoebot. Let the *amoebot structure* $X \subseteq V$ be the set of nodes occupied by the amoebots. We assume that $G_X = (X, E_X)$ is connected, where $G_X = G_\Delta|_X$ is the graph induced by X. In the geometric variant of the model, G is the infinite regular triangular grid graph $G_\Delta = (V_\Delta, E_\Delta)$ (see Figure 1a).

3 Reconfigurable Circuit Extension

In the *reconfigurable circuit extension* [8], each edge between two neighboring amoebots u and v is replaced by c edges called *external links* with endpoints called *pins*, for some constant $c \geq 1$ that is the same for all amoebots. For each of these links, one pin is owned by u while the other pin is owned by v. In this paper, we assume that neighboring amoebots have a common labeling of their incident external links.

Each amoebot u *partitions* its *pin set* $PS(u)$ into a collection $\mathcal{C}(u)$ of pairwise disjoint subsets such that the union equals the pin set, i.e., $PS(u) = \bigcup_{C \in \mathcal{C}(u)} C$. We call $\mathcal{C}(u)$ the *pin configuration* of u and $C \in \mathcal{C}(u)$ a *partition set* of u. Let $\mathcal{C} = \bigcup_{u \in S} \mathcal{C}(u)$ be the collection of all partition sets in the system. Two partition sets are *connected* iff there is at least one external link between those sets. Let L be the set of all connections between the partition sets in the system. Then, we call $H = (\mathcal{C}, L)$ the *pin configuration* of the system and any connected component C of H a *circuit* (see Figure 1b). Note that if each partition set of $\mathcal{C}$ is a *singleton*, i.e., a set with exactly one element, then every circuit of H just connects two neighboring amoebots. An amoebot is part of a circuit iff the circuit contains at least one of its partition sets. A priori, an amoebot u may not know whether two of its partition sets belong to the same circuit or not since initially it only knows $\mathcal{C}(u)$.

Each amoebot u can send a primitive signal (a *beep*) via any of its partition sets $C \in \mathcal{C}(u)$ that is received by all partition sets of the circuit containing C at the beginning of the next round. The amoebots are able to distinguish between beeps arriving at different partition sets. More specifically, an amoebot receives a beep at partition set C if at least one amoebot sends a beep on the circuit belonging to C, but the amoebots neither know the origin of the signal nor the number of origins.

We assume the fully synchronous activation model, i.e., the time is divided into synchronous rounds, and every amoebot is active in each round. On activation, each amoebot may update its state, reconfigure its pin configuration, and activate an arbitrary number of its partition sets. The beeps are propagated on the updated pin configurations. The time complexity of an algorithm is measured by the number of synchronized rounds required by it.

4 Problem Statement and Our Contribution

Let $S, D \subseteq X$ be two non-empty subsets. We call each amoebot in S a *source*, and each amoebot in D a *destination*. A (S, D)*-shortest path forest* is a set of rooted trees that satisfies the following properties.

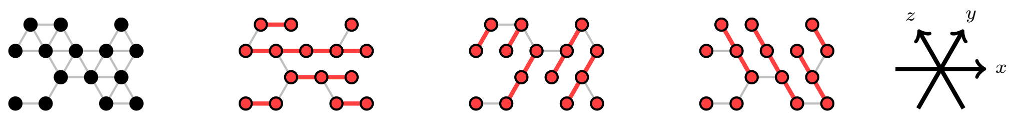
1. For each $s \in S$, the set contains a tree $T_s = (V_s, E_s)$ rooted at s with $V_s \subseteq X$ and $E_s \subseteq E_X$.
2. For each $s \in S$, each leaf of T_s is in $S \cup D$.
3. For each $s_1, s_2 \in S$, V_{s_1} and V_{s_2} are disjoint.
4. For each $u \in D$, there is a tree T_s such that $u \in V_s$, i.e., $D \subseteq \bigcup_{s \in S} V_s$.
5. For each $s \in S$ and $u \in V_s$, the unique path from s to u in T_s is a shortest path from s to u in G_X, and s has the smallest distance to u among all amoebots in S.

We call an (S, X)-shortest path forest also an S*-shortest path forest.*

We consider the (k, ℓ)*-shortest path forest problem* $((k, \ell)$*-SPF)* for $k, \ell \geq 1$. Let two sets $S, D \subseteq X$ of amoebots be given such that $|S| = k$ and $|D| = \ell$, i.e., each amoebot $u \in X$ knows whether $u \in S$ and whether $u \in D$. We say that X computes a (S, D)-shortest path forest if each amoebot in $\bigcup_{s \in S} V_s \setminus S$ knows its parent within the (S, D)-shortest path forest. The goal of the amoebot structure is to compute a (S, D)-shortest path forest.

Note that we obtain the classical *single pair shortest path problem (SPSP)* for $k = \ell = 1$, and the classical *single source shortest path problem (SSSP)* for $k = 1$ and $\ell = n$.

We will present two deterministic algorithms for (k, ℓ)-SPF. For hole-free amoebot structures, our *shortest path tree algorithm* solves the problem within $O(\log \ell)$ rounds for $k = 1$, and our *shortest path forest algorithm* within $O(\log n \log^2 k)$ for $k \geq 1$. Note that the former result implies that we can solve SPSP within $O(1)$ rounds, and SSSP within $O(\log n)$ rounds.



(a) Initial structure. **(b)** x-portal graph. **(c)** y-portal graph. **(d)** z-portal graph. **(e)** Axes.

Figure 2 Portal graphs. Each red connected component indicates a portal. We obtain the portal graphs by fusing the amoebots of each portal to a single node.

5 Related Work

The reconfigurable circuit extension was introduced by Feldmann et al. [8]. They proposed solutions for the leader election, compass alignment, and chirality agreement problems. Each of these solutions requires $O(\log n)$ rounds w.h.p. Afterwards, they considered the recognition of various classes of shapes. An amoebot structure is able to detect parallelograms with linear or polynomial side ratio within $O(\log n)$ rounds w.h.p. Further, an amoebot structure is able to detect shapes composed of triangles within $O(1)$ rounds if the amoebots agree on a chiracilty. Feldmann et al. proposed the PASC algorithm which allows the amoebot structure to compute distances along a path [8, 12]. With the help of it, Padalkin et al. were able to solve the global maxima, spanning tree, and symmetry detection problems in polylogarithmic time [12].

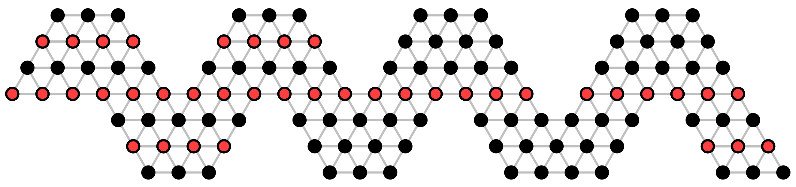
Shortest path problems are broadly studied both in the sequential and distributed setting. In distributed setting, adjacent nodes are able to communicate via messages. The CONGEST model limits the size of each message to a logarithmic number of bits (in n). For weighted SSSP, Chechik and Mukhtar proposed a randomized algorithm that takes $\tilde{O}(\sqrt{n}d^{1/4} + d)$ rounds [1]. The best known lower bound is $\Omega(\sqrt{n} + d)$ [7, 13].

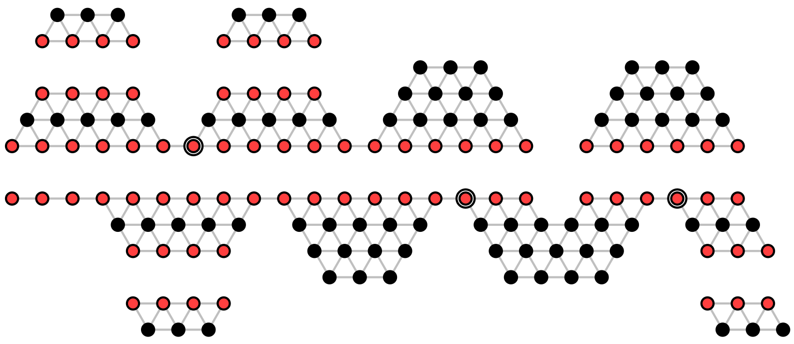
In the amoebot model, Kostitsyna et al. were the first to consider SSSP [9, 10]. By applying a breadth-first search approach, they compute a shortest path tree within $O(n^2)$ rounds. For simple amoebot structures without holes, they introduced feather trees – a special type of shortest path trees. These can be computed within $O(d)$ rounds where d is the diameter of the structure. To our knowledge, there is no further work on shortest path problems in the amoebot model or its reconfigurable circuit extension.

6 Results

Coy et al. [3] have solved the shortest path problem for hybrid communication networks that can be modelled as grid graphs without holes. Our shortest path tree algorithm for $k = 1$ adapts their approach as follows. The idea is to utilize portal graphs (see Figure 2). The vertices (called *d-portals*) of the *d-portal graph* are the connected components if we remove all edges that are not in parallel with the d-axis. Two d-portals are adjacent in the d-portal graph iff there exists an edge between them. We can show that for triangular grid graphs without holes, the distance between two nodes is half the sum of the distances of their portals in the portal graphs. Note that this equation does not hold anymore if the amoebot structure has holes. Further, it can be shown that the portal graph is a tree. This allows us to utilize the Euler tour technique [14] in combination with the PASC algorithm to compute distances in the portal graphs. We obtain the following result.

▶ **Theorem 6.1.** *The shortest path tree algorithm computes an $(\{s\}, D)$-shortest path forest within $O(\log \ell)$ rounds.*


(a) Initial amoebot structure.


(b) Regions after the first phase.

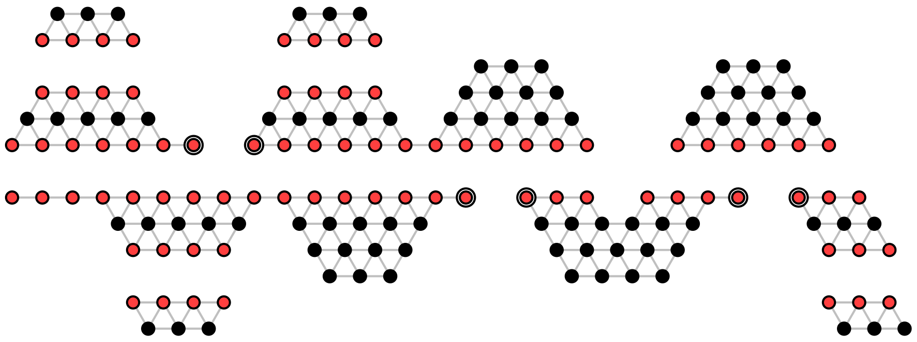

(c) Regions after the second phase.

Figure 3 Regions. The red amoebots indicate the splitting portals. The encircled amoebots indicate the splitting amoebots.

Our shortest path forest algorithm for $k \geq 1$ takes a divide and conquer approach. The idea is to split the amoebot structure into two regions, to recursively compute a shortest path forest for both regions, and to finally merge them. In the following, we will elaborate on these steps.

In the first step, we split the amoebot structure into smaller regions. For that, we make use of so-called *x-portals* [2], i.e., connected components of the intersections of the amoebot structure with an x-axis (see red amoebots in Figure 3a). Our goal is to split the amoebot structure until each region intersects at most two x-portals with at least one source. We split the amoebot structure in two phases. In the first phase, we split the amoebot structure at all portals with at least one source and at most Ok further portals (see Figure 3b). The splitting portal is part of both regions. In the second step, we split the amoebot structure further at amoebots at bottlenecks (see Figure 3c). The splitting amoebot is part of both regions. We can easily compute a shortest path forest for each resulting region. We omit the details for these computations. Each region will maintain a shortest path forest.

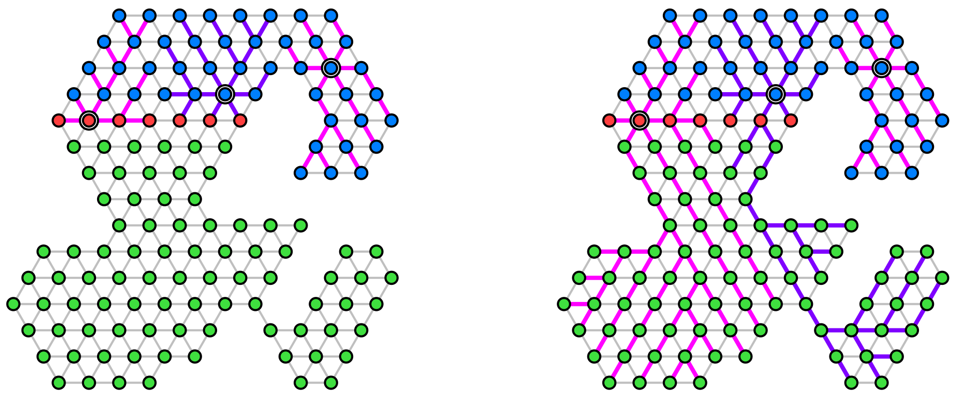


Figure 4 Propagation of the shortest path forest of the blue region into the green region. The left figure shows the initial situation, and the right figure the result. The red amoebots indicate the portal. The encircled amoebots indicate the sources. The pink and purple edges indicate the shortest path trees.

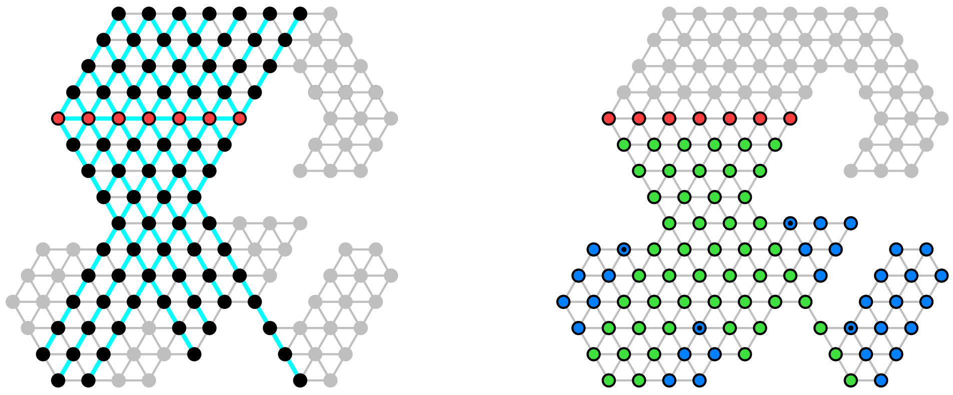


Figure 5 Phases of the propagation procedure. The red amoebots indicate the portal. The left figure indicates the amoebots visible by the portal. The right figure shows the phases. In the first phase, we propagate the shortest path forest to the green amoebots. In the second phase, we propagate the shortest path forest to the blue amoebots. We propagate the shortest path forests through the amoebots with a dot.

In order to merge two adjacent regions and with that their shortest path forests, we proceed as follows. We first propagate the shortest path forests of both regions into the other region, respectively (see Figure 4). The propagation happens in two phases. In the first phase, we propagate the shortest path forest to all amoebots *visible* by the splitting portal (see green amoebots in Figure 5). We can show that each amoebot can determine its parent by comparing the distances of two amoebots on the portal to their closest sources. In the second phase, we propagate the shortest path forest to the remaining amoebots, which may form several connected components. We propagate the shortest path forest into each of those independently of each other. For each connected component, we can propagate the shortest path forest through one of the its amoebots (see amoebots with a dot in Figure 5). This allows us to apply our shortest path tree algorithm for $k = 1$ with that amoebot as the source to finish the propagation (see Theorem 6.1).

Finally, we have to merge the resulting shortest path forests of both regions. By applying the PASC algorithm on each path from the source to a leaf in both shortest path forests, each amoebots computes its distance to the closest source of both regions. This allows each amoebot to determine the closest source of both regions and with that to choose its parent in the merged shortest path forest.

Note that all steps (splitting, propagation, merging, etc.) make extensive use of the geometric properties of the amoebot structure, e.g., that it is free of holes. By utilizing $O(k)$ portals to split the amoebot structure into regions and by merging the regions with respect to a centroid decomposition tree, we obtain the following result.

▶ **Theorem 6.2.** *The shortest path forest algorithm computes an (S, D)-shortest path forest within $O(\log n \log^2 k)$ rounds.*

References

1 Shiri Chechik and Doron Mukhtar. Single-source shortest paths in the CONGEST model with improved bounds. *Distributed Comput.*, 35(4):357–374, 2022.

2 Sam Coy, Artur Czumaj, Michael Feldmann, Kristian Hinnenthal, Fabian Kuhn, Christian Scheideler, Philipp Schneider, and Martijn Struijs. Near-shortest path routing in hybrid communication networks. In *OPODIS*, volume 217 of *LIPIcs*, pages 11:1–11:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

3 Sam Coy, Artur Czumaj, Christian Scheideler, Philipp Schneider, and Julian Werthmann. Routing schemes for hybrid communication networks. In *SIROCCO*, volume 13892 of *Lecture Notes in Computer Science*, pages 317–338. Springer, 2023.

4 Joshua J. Daymude, Andréa W. Richa, and Jamison W. Weber. Bio-inspired energy distribution for programmable matter. In *ICDCN*, pages 86–95. ACM, 2021.

5 Zahra Derakhshandeh, Shlomi Dolev, Robert Gmyr, Andréa W. Richa, Christian Scheideler, and Thim Strothmann. Brief announcement: amoebot - a new model for programmable matter. In *SPAA*, pages 220–222. ACM, 2014.

6 Zahra Derakhshandeh, Robert Gmyr, Andréa W. Richa, Christian Scheideler, and Thim Strothmann. Universal shape formation for programmable matter. In *SPAA*, pages 289–299. ACM, 2016.

7 Michael Elkin. An unconditional lower bound on the time-approximation trade-off for the distributed minimum spanning tree problem. *SIAM J. Comput.*, 36(2):433–456, 2006.

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

9 Irina Kostitsyna, Tom Peters, and Bettina Speckmann. Brief announcement: An effective geometric communication structure for programmable matter. In *DISC*, volume 246 of *LIPIcs*, pages 47:1–47:3. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

10 Irina Kostitsyna, Tom Peters, and Bettina Speckmann. Fast reconfiguration for programmable matter. In *DISC*, volume 281 of *LIPIcs*, pages 27:1–27:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.

11 Giuseppe Antonio Di Luna, Paola Flocchini, Nicola Santoro, Giovanni Viglietta, and Yukiko Yamauchi. Shape formation by programmable particles. *Distributed Comput.*, 33(1):69–101, 2020.

12 Andreas Padalkin, Christian Scheideler, and Daniel Warner. The structural power of reconfigurable circuits in the amoebot model. In *DNA*, volume 238 of *LIPIcs*, pages 8:1–8:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

13 Atish Das Sarma, Stephan Holzer, Liah Kor, Amos Korman, Danupon Nanongkai, Gopal Pandurangan, David Peleg, and Roger Wattenhofer. Distributed verification and hardness of distributed approximation. *SIAM J. Comput.*, 41(5):1235–1265, 2012.

14 Robert Endre Tarjan and Uzi Vishkin. An efficient parallel biconnectivity algorithm. *SIAM J. Comput.*, 14(4):862–874, 1985.

15 Tommaso Toffoli and Norman Margolus. Programmable matter: Concepts and realization. *Int. J. High Speed Comput.*, 5(2):155–170, 1993.

16 Jamison W. Weber, Tishya Chhabra, Andréa W. Richa, and Joshua J. Daymude. Energy-constrained programmable matter under unfair adversaries. *CoRR*, abs/2309.04898, 2023.