Fault Tolerant Shape Formation in the Amoebot Model

Christian Scheideler
Paderborn University, Department of Computer Science, Paderborn, Germany

Daniel Warner
Paderborn University, Department of Computer Science, Paderborn, Germany

Abstract

The amoebot model is a distributed computing model of programmable matter. It envisions programmable matter as a collection of computational units called amoebots or particles that utilize local interactions to achieve tasks of coordination, movement and conformation. In the geometric amoebot model the particles operate on a hexagonal tessellation of the plane. Within this model, numerous problems such as leader election, shape formation or object coating have been studied. One area that has not received much attention so far, but is highly relevant for a practical implementation of programmable matter, is fault-tolerance. The existing literature on that aspect allows particles to crash but assumes that crashed particles do not recover. We propose a new model in which a crash causes the memory of a particle to be reset and a crashed particle can detect that it has crashed and try to recover using its local information and communication capabilities. We propose an algorithm that solves the hexagon shape formation problem in our model if a finite number of crashes occur and a designated leader particle does not fail. At the heart of our solution lies a fault-tolerant implementation of the spanning forest primitive, which, since other algorithms in the amoebot model also make use of it, is also of general interest.

Model extension In our work we extend the *geometric amoebot model* (introduced in [1]) by the aspect of fault-tolerance and, in order to gain initial insights, focus on the problem of shape formation using the hexagon shape formation problem as basis. We extend the model by introducing *particle crashes*. We assume that the adversarial scheduler may arbitrarily crash particles. A crash of a particle p has the following effects: The scheduler sets the *state* in p's local memory to CRASHED and may arbitrarily change the rest of p's local memory. This means that p and its neighbours can reliably detect that it has crashed, the rest of p's local memory, however, is no longer reliable. Particles can then try to recover using their local communication capabilities.

Problem description For any two nodes $u, v \in V_\triangle$ the *distance* $\delta(u,v) \in \mathbb{N}_0$ between u and v is defined as the length of a shortest path from u to v in $G_\triangle$. For a node $v \in V_\triangle$ and $i \in \mathbb{N}_0$ let $B(v,i) := \{\, u \in V_\triangle \mid \delta(u,v) = i \,\}$. We call a set $V \subseteq V_\triangle$ a *hexagon with centre* $v \in V$ if there $k \in \mathbb{N}_0$ and a subset $S \subseteq B(v,k)$ such that $V = S \cup \bigcup_{i<k} B(v,i)$. We define the *hexagon shape formation problem* **HEX**: We assume that the system of particles initially forms a single connected component of contracted particles, has a unique leader, called the SEED particle, and that all other particles are IDLE. The goal is to reach a stable configuration in which the set of nodes occupied by particles is a hexagon with the SEED in its centre.

Main results We propose an algorithm `HexagonFT` that solves the hexagon shape formation problem **HEX** in our model under the presence of particle crashes. Our two main results are:

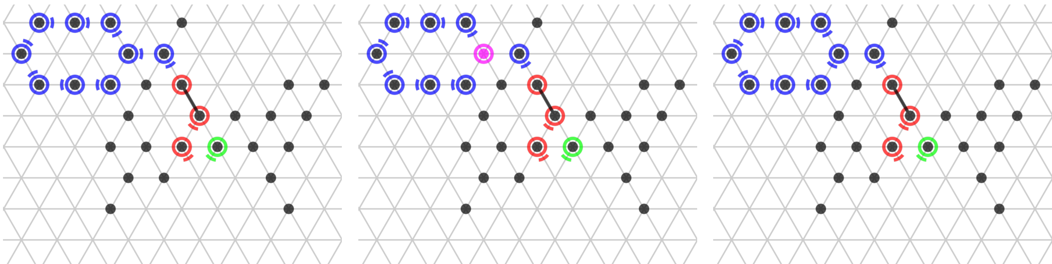
▶ **Lemma 1.** *If a finite number of crashes occur during the execution of algorithm* `HexagonFT` *and m particles are faulty after the last crash, then a non-faulty configuration is reached within $\mathcal{O}(mn)$ rounds after the last crash.*

▶ **Theorem 2.** *If a finite number of crashes occur, then the algorithm* `HexagonFT` *solves the hexagon shape formation problem* **HEX** *in worst-case $\mathcal{O}(n^2)$ work (total number of moves*

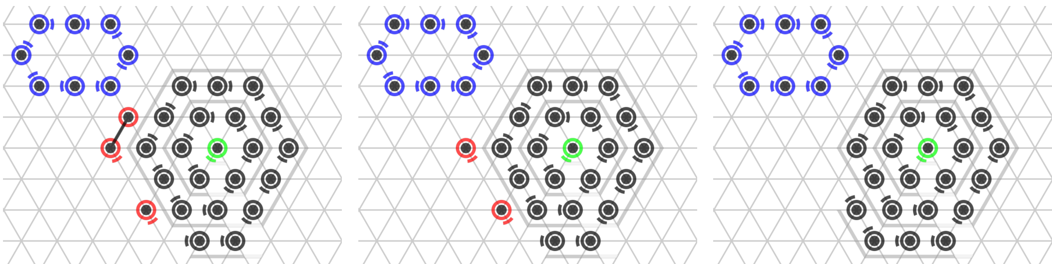
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

executed by all particles). From the time when no more crashes occur and the configuration is non-faulty, the algorithm needs $\mathcal{O}(n)$ rounds until termination.

Illustration Our fault-tolerant solution builds upon two primitives, a *propagation primitive* and a *safety primitive*, each of which solves a significant problem: Firstly, we need to ensure that when a crashed particle chooses a FOLLOWER as parent, this does not lead to disconnection of the particles (see Figure 1). In order to avoid disconnection, we use a validation mechanism that determines for a faulty particle which of the FOLLOWER parent candidates it can attach to without closing a cycle. Secondly, we must ensure that particles within the hexagon shape formed so far do not leave it at any time. If this is not ensured, various problematic consequences are possible: among other things, the particles may be disconnected, it is not guaranteed that a hexagon will be built, and it is even possible that the algorithm will no longer terminate.



(a) Legal configuration **(b)** A particle has crashed. **(c)** The crashed particle attaches itself to the wrong of the two possible FOLLOWER candidates, closing a cycle.



(d) Just before the particles get disconnected **(e)** Particles are disconnected **(f)** Final stable configuration

Figure 1 The figure shows that if a crashed particle attaches itself to a arbitrary FOLLOWER pointing away from it, this can lead to irreversible disconnection of the particles.

References

1. Zahra Derakhshandeh, Shlomi Dolev, Robert Gmyr, Andréa W Richa, Christian Scheideler, and Thim Strothmann. Brief announcement: amoebot–a new model for programmable matter. In *Proceedings of the 26th ACM symposium on Parallelism in algorithms and architectures*, pages 220–222, 2014. 1