

Distributed leader election and computation of local identifiers for programmable matter

Nicolas Gastineau, Wahabou Abdou, Nader Mbarek, Olivier Togni

▶ **To cite this version:**

Nicolas Gastineau, Wahabou Abdou, Nader Mbarek, Olivier Togni. Distributed leader election and computation of local identifiers for programmable matter. Algorithms for Sensor Systems, 11410, Springer Nature, pp.159-179, 2019, Lecture Notes in Computer Science, 978-3-030-14094-6. hal-01848911

HAL Id: hal-01848911
https://hal.science/hal-01848911

Submitted on 25 Jul 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed leader election and computation of local identifiers for programmable matter

Nicolas Gastineau[1], Wahabou Abdou[1], Nader Mbarek[1], and Olivier Togni[1]

[1]LE2I FRE2005, CNRS, Arts et Métiers, Université Bourgogne Franche-Comté, F-21000 Dijon, France

July 25, 2018

Abstract

The context of this paper is programmable matter, which consists of a set of computational elements, called *particles*, in an infinite graph. The considered infinite graphs are the square, triangular and king grids. Each particle occupies one vertex, can communicate with the adjacent particles, has the same clockwise direction and knows the local positions of neighborhood particles. Under these assumptions, we describe a new leader election algorithm affecting a variable to the particles, called the k-local identifier, in such a way that particles at close distance have each a different k-local identifier. For all the presented algorithms, the particles only need a $O(1)$-memory space.

1 Introduction

Programmable matter can be seen as modular robots (called modules or particles) able to fix to adjacent modules and send (receive) messages to (from) other modules fixed to the entity. Thus, the different modules form a geometric shape which is a network. Usually, a module can fix to another module using a finite number of ports (see Figure 1 for an example of spherical modules). Also, the modules know the ports that are in contact with other modules and have a knowledge about the geographic position of their ports. Moreover, the ports are supposed to be homogeneously distributed along the surface of each module. Such assumptions imply that the way how the modules are on a plane can be modeled by a grid. In this paper, we only consider modules on a plane surface, i.e. two dimensional grids. In this context, the geometric amoebot model [6, 7, 8, 9, 10, 11] aims to model the properties of a network for programmable matter.

Distributed algorithms aim to give a theoretical algorithmic framework in order to model the execution of an algorithm that runs on a network of computational elements that can cooperate in order to solve network problems. In distributed algorithm frameworks, it is often supposed that the different elements of the network do not have a unique identity, i.e., the network is anonymous. In anonymous networks, a natural question is how to perform a leader election, i.e., how to determine a singular element in an anonymous network. It is well known that for some network structures, the ring for example, there is no deterministic leader election algorithm [1].

In 1999, Antoni Mazurkiewicz [19] has presented a deterministic general algorithm to determine a leader (in the case it is possible to do so). In the situation where the elements have access to a random source, then it is also proven that no algorithm can correctly determine a leader in a ring with any probability $\alpha > 0$ [15]. Due to the assumption we make about the ports of the particles in the context of programmable matter (a particle knows the ports which are in contacts with other particles and knows the geographic position of its ports), the leader election problem becomes different than in the classical system. In particular, in the field of programmable matter,

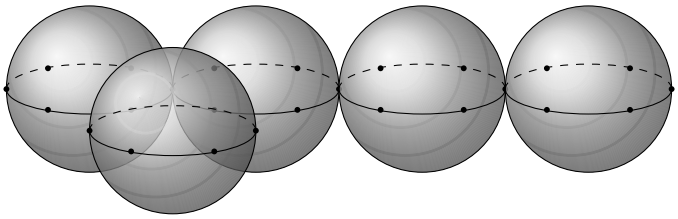


Figure 1: Five spherical particles forming a simple structure (circle: port of the particles).

there exists a probabilistic algorithm that determine a leader (and in particular for a ring) with probability 1 [5].

Several projects aim to build programmable matter prototypes. One of such projects [20, 23], financed by the french National Agency for Research, aims to build cuboctahedral particles able to deform them-selves in order to move. This project can be split in two phases, one consists in manufacturing the hardware of prototype matters, the second consists in proposing algorithms for programmable matter. The final goal of this project is to sculpt a shape-memory polymer sheet with programmable matter. In the continuity of the algorithm phase of this project [20], we propose algorithms for the self-configuration, i.e., in order to create identifiers and spanning trees.

In the context of programmable matter [3, 4, 14, 18, 23, 24], it is supposed that a network can contain several millions of modules and that each module has possibly a nano-centimeter size. These two facts lead us to believe that even a $O(\log(n))$-space memory for each module, n being the number of modules, is not technically possible. Also, because of the large number of modules, it can be very challenging and time consuming to implement a unique identity to the modules when they are created. In this context, we suppose that the modules can not store a unique identity, i.e., that the network is anonymous. In this paper we propose deterministic $O(1)$-space memory algorithms to determine a leader in the network and to create k-local identifiers of the particles. A k-local identifier is a variable affected to each module of the network which is different for every two modules at distance at most k. Note that leader election [5, 13] plays a significant role in numerous problems of programmable matter.

Our contribution is the following: we introduce a leader election algorithm based on local computations and simple to implement. This algorithm works when the structure the particles form has no hole (see Section 3). Also, since the algorithm can be described as a sequence of local computations, its limits (message complexity, required memory-space, etc) are easy to analyze. We present a distributed algorithm to construct a spanning tree in the context of programmable matter and, also, a distributed algorithm to re-organize the port numbers of the particles. Finally, we present an algorithm to assign a k-local identifier to each particle. In order to compute k-local identifiers, we suppose that we have done a leader election before. The k-local identifiers are determined using graph theoretical results about the coloring of the k^{th} power of the grids. An advantage of the given k-local identifiers is that they are really simple to update in case the particles move and, consequently, the structure that the particles form changes.

This paper is organized as follows: in Section 2, we present our algorithmic framework in the context of distributed algorithms for programmable network. In the third section, we present our leader election algorithm. Finally, in Section 4, we present our algorithm to assign k-local identifiers to the particles (using the colorings from Appendix I).

2 Notation, definitions and our programmable matter algorithmic framework

The geometric amoebot model [6, 7, 8, 9, 10, 11] aims to model the computations that can occur in the context of programmable matter. In this paper, we use an algorithmic framework inspired by the geometric amoebot model. We assume that any structure the different particles can form

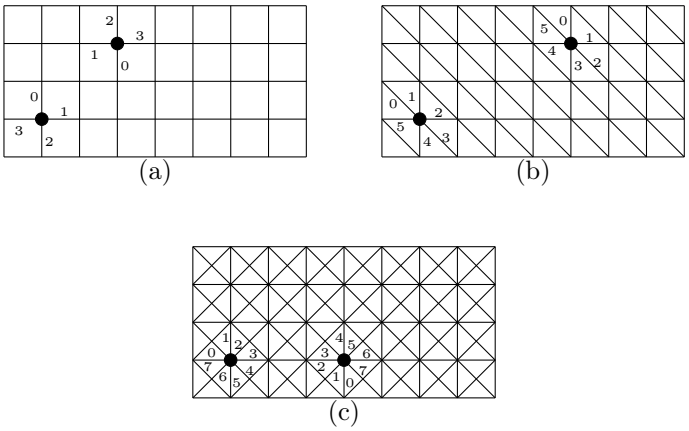


Figure 2: Subgraphs of the square (Figure 2.a), triangular (Figure 2.b) and king (Figure 2.c) grids, with the port numbers of two particles.

is a subgraph of an infinite graph G. In this graph, $V(G)$ represents all possible positions the particles can occupy and $E(G)$ represents possible connections between particles. The set $E(G)$ also represents the possible movements from a position to another position (for a particle). We suppose that two particles can bond each other, i.e., can communicate only in the case they are on adjacent positions. The two following paragraphs are dedicated to the notation and definitions we use for graphs.

For a graph G, we denote by $V(G)$ the *vertex set* of G and by $E(G) \subseteq V(G) \times V(G)$ the *edge set* of G. We denote by $d_G(u, v)$, the usual distance between two vertices u and v in G. If we consider the distance in a subgraph H of G, the distance will be denoted by $d_H(u, v)$. The *diameter* of G, denoted by $\text{diam}(G)$, is $\max(\{d_G(u, v)|\ u, v \in V(G)\})$. The set $N_G(u) = \{v \in V(G)|\ uv \in E(G)\}$ is the set of *neighbors* of u. By $\Delta(G)$, we denote the *maximum degree* in G, i.e., the maximum cardinality of $N_G(u)$, for $u \in V(G)$. Finally, we denote by $G[S]$, for $S \subseteq V(G)$, the subgaph induced by the vertices from S and by $G - S$ the subgraph of G induced by the vertices from $V(G) \setminus S$.

In the remaining part of this paper, the graphs considered will be the infinite *square*, *triangular* and *king* grids. We denote by $\mathcal{S}$ the square grid, by $\mathcal{T}$ the triangular grid and by $\mathcal{K}$ the king grid. A subgraph of each of these three infinite graphs is represented in Figure 2. Moreover, we suppose that these three grids are represented on a plane as in Figure 2. For these grids, the considered vertex set is $\{(i, j)|\ i, j \in \mathbb{Z}\}$ and the edge sets are the following:

- $E(\mathcal{S}) = \{(i, j)(i \pm 1, j)|\ i, j \in \mathbb{Z}\} \cup \{(i, j)(i, j \pm 1)|\ i, j \in \mathbb{Z}\}$;
- $E(\mathcal{T}) = E(\mathscr{S}) \cup \{(i, j)(i + 1, j - 1)|\ i, j \in \mathbb{Z}\} \cup \{(i, j)(i - 1, j + 1)|\ i, j \in \mathbb{Z}\}$;
- $E(\mathcal{K}) = E(\mathscr{T}) \cup \{(i, j)(i + 1, j + 1)|\ i, j \in \mathbb{Z}\} \cup \{(i, j)(i - 1, j - 1)|\ i, j \in \mathbb{Z}\}$.

We also remind the distance between two vertices (i, j) and (i', j') in the three different grids:

- $d_{\mathcal{S}}((i, j), (i', j')) = |i - i'| + |j - j'|$;
- $d_{\mathcal{T}}((i, j), (i', j')) = \begin{cases} \max(|i - i'|, |j - j'|), & \text{if } (i \geq i' \wedge j \leq j') \vee (i \leq i' \wedge j \geq j'); \\ |i - i'| + |j - j'|, & \text{otherwise;} \end{cases}$
- $d_{\mathcal{K}}((i, j), (i', j')) = \max(|i - i'|, |j - j'|)$.

Note that there is a way to draw the triangular grid in which each triangle is equilateral. However, we prefer to draw it as a subgraph of the king grid (see Figure 2) in order to have illustrations for which the vertex set $\{(i, j)|\ i, j \in \mathbb{Z}\}$ corresponds to the position of the vertices in the plane.

In both representation, the notion of distance coincide but is easier to observe in our chosen representation. However, note that the representation of the triangular grid in which each triangle is equilateral corresponds to the optimal way to pack unit disks in the plane (the position of the vertices in this representation corresponds to the center of the unit disk and an edge represents a contact between two disks).

We also denote by $i \pmod p$ or $i \ _{(\mathrm{mod}\ p)}$, depending on the context, the integer j such that $j \equiv i \pmod p$ and $0 \leq j < p$. The remaining part of this subsection is dedicated to our programmable matter algorithmic framework.

We give the following properties about the particles and vertices of the graph:

- each particle occupies a single vertex and each vertex is occupied by at most one particle;
- the subgraph induced by the occupied vertices is supposed to be connected.

The subgraph induced by the occupied vertices of $V(G)$ is called the *particle graph* and is denoted by P. The vertex occupied by a particle p is denoted by $s(p)$. For a particle p, $N_G(p) = \{u \in V(G) |\ u \in N_G(s(p))\}$. The *ports* of a particle are the endpoints of communication. Each particle has $\Delta(G)$ ports in a regular grid G ($\Delta(G) = 4$ for $G = \mathcal{S}$, $\Delta(G) = 6$ for $G = \mathcal{T}$ and $\Delta(G) = 8$ for $G = \mathcal{K}$). The ports of a particle occupying a vertex u are represented by the edges incident with u. An edge between two vertices represents a possible communication between two particles p_1 and p_2 occupying these two vertices using each one a different port. A particle has the following properties:

- each particle is anonymous, i.e., it does not have an identifier;
- each particle has a collection of ports, each labeled by a different integer from $\{0, \ldots, \Delta(G) - 1\}$;
- the port numbers are given as a function of the position of the edges on a plane representation of the grids (see Figure 2);
- each particle knows the labels of the ports that can communicate with particles from the neighborhood;
- each particle knows the state of the neighbors.

In our algorithmic framework, we suppose that the particles have their ports labeled following the same clockwise order. Thus, consecutive port numbers correspond to consecutive edges around a vertex (as in the representation on the plane from Figure 2). Note that the particles do not have the same notion of orientation, i.e., there is possibly not a unique label for ports that correspond to edges going in the same cardinal direction. In the presented algorithms, the state of a particle will contain a variable corresponding to the status of the particle in the leader election algorithm and the information regarding its parents and childs for a constructed spanning tree.

The proposed algorithms in our algorithmic framework are results of successive local computations [2, 21]. In particular, the first presented leader election algorithm from Section 3 can be described by a graph relabeling system [2] which is a local computation system. In this paper, the correct execution of the different algorithms is only guaranteed if the algorithms are ran in the order depicted in Figure 3.

We suppose the following:

- each particle contains the same program and begins in the same state;
- the computation process is represented by successive local computations;
- no local computation occurs simultaneously on two particles at distance at most 2;
- during a local computation, a particle can perform a bounded number of computations and can send messages to its neighbors;

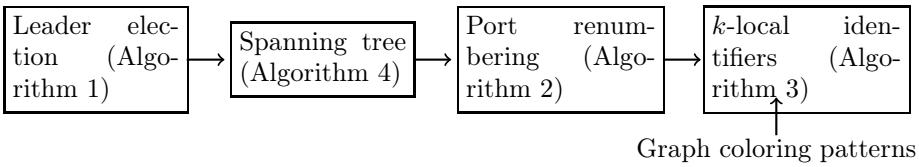


Figure 3: An illustration of the algorithm dependency (arrow between algorithms/results: dependency of one algorithm to another algorithm/result).

- a *round* is a sequence of successive local computations for which each particle does at least one local computation;
- an algorithm finishes in k rounds if after any k successive rounds the algorithm is finished.

Note that the concept of rounds is used to bound the running time of the algorithms. In our algorithm framework we suppose that no two particles at distance at most 2 perform computations simultaneously in order to simplify the presentation of our results. However, this supposition can be removed by implementing, for example, a probabilistic leader election algorithm on the vertices at distance at most 2 of one of the two vertices, i.e., by computing a random value on the vertices at distance 2 and doing the local computation following the increasing order of the values. In order to compute the running time of an algorithm in case of a specific programmable matter prototype, the complexity of the algorithm should be computed using the required number of rounds and the required running time in order to avoid that two particles at distance at most 2 perform computations simultaneously.

3 Leader election

In this section we present a new leader election algorithm. This algorithm is very easy to implement but requires that the particle graph has a specific structure. In this algorithm, the required memory space is constant, the messages have constant size, the required computation power of the particle has been optimized and the required number of rounds is less than $2n$ (n being the number of particles).

A *hole* in a subgraph G' of a graph G among the three grids is a subgraph H of G satisfying three properties:

i) $V(H)$ is finite, H is connected and $|V(H)| \geq 1$;

ii) $V(H) \cap V(G') = \emptyset$;

iii) every vertex $u \in V(H)$ satisfies $N_G(u) \subseteq V(H) \cup V(G')$.

Less formally, a subgraph G' of one of the three grids contains a hole if there is a finite space only containing vertices from $V(G) \setminus V(G')$ which are surrounded by vertices of G'. A hole containing three vertices is illustrated in the left part of Figure 4. We call G' *hole-free*, when G' has no holes.

If the particle graph P on G is hole-free, then every particle p which satisfies $|N_G(p) \cap V(P))| < \Delta(G)$ is at the geographical border of the shape of P. Moreover, we call the set of particles p which satisfy $|N_G(p) \cap V(P))| < \Delta(G)$ and such that the vertices $N_G(p) - V(P)$ are not all in a hole of P, the *border* of P. The right part of Figure 4 illustrates the border of P.

In addition, for a particle p occupying a vertex (i, j) of the square grid, the four vertices $(i + 1, j + 1)$, $(i - 1, j + 1)$, $(i + 1, j - 1)$ and $(i - 1, j - 1)$ are the *corners* of p and the set of corners is denoted by $C(p)$. The *extended neighborhood* of a particle p, denoted by $M_G(p)$, is the set $N_G(p)$ if G is the triangular grid or king grid or the set $N_G(p) \cup C(p)$ if G is the square grid. Note that we define the extended neighborhood differently for the square grid in order to be able to present a generic algorithm (Algorithm 1) that works for all the three grids.

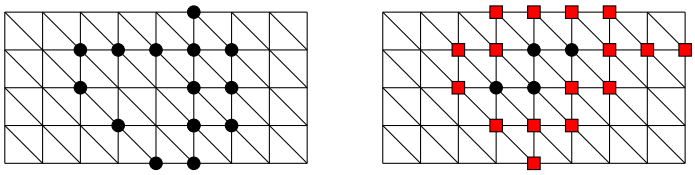


Figure 4: A hole in P (on the left) and the border of P in the case P is hole-free (on the right; square: particle on the border of P).

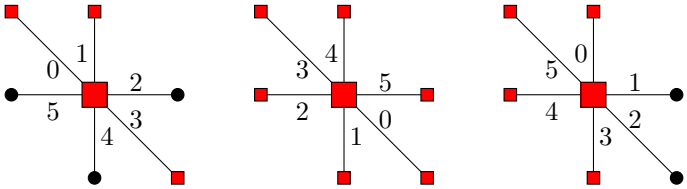


Figure 5: Two non S-contractible particles (at the center of the left and the middle drawing) and an S-contractible particle (at the center of the right drawing) in the triangular grid (square: particle in S; circle: particle not in S).

We give the following definition of S-contractible particle (see Figure 5) that will be used in our leader election algorithm.

Definition 1. *Let G be an infinite grid among $\mathcal{S}$, $\mathcal{T}$ and $\mathcal{K}$ and let $S \subseteq V(P)$, for P the particle graph on G. A particle p is said to be S-contractible if it satisfies the following properties:*

I) $G[M_G(p) \cap S]$ is connected;

II) $|N_G(p) \cap S| < \Delta(G)$, i.e., there exists a neighbor of p in G which is not occupied by a particle from S.

A particle p is an *articulation* of a connected subgraph G' of one of the three grids if $G' - \{s(p)\}$ is not connected. Derakhshandeh et al. [8] proposed a randomized leader election algorithm in the geometric amoebot model in the case there is no particle which is an articulation. Our proposed leader election algorithm (Algorithm 1) works even if $V(P)$ contains a particle p which is an articulation. However, in contrast with the leader election algorithm from Derakhshandeh et al. [8], Algorithm 1 does not work if P has holes. In the remaining part of this paper, Algorithm 1 is called the S-contraction algorithm.

Recently, Daymude et al. [5] have improved the algorithm from Derakhshandeh et al. [8] in order that it works when $V(P)$ contains an articulation. However, it remains challenging to implement it.

Also, very recently, Di Luna et al. [13] have introduced a leader election algorithm called consumption algorithm. The consumption and the S-contraction algorithms both consist in successively removing the candidacy of the particles on the border of P. However, one can easily notice that, in our algorithm, we possibly remove the candidacy of particles having four or five neighbors (which is not considered in the consumption algorithm). Also, the consumption algorithm does not work on square and king grids and the considered theoretical frameworks for the two algorithms are different.

In the S-contraction algorithm (Algorithm 1), the particles can be in three different states: **C** (candidate), **N** (not elected) and **L** (leader). We suppose that every particle begins in the state **C**.

Let S be the particle in state **C**. Algorithm 1 consists in removing from S the particles which are both on the border of $G[S]$ and not articulations of $G[S]$. An example of the execution of Algorithm

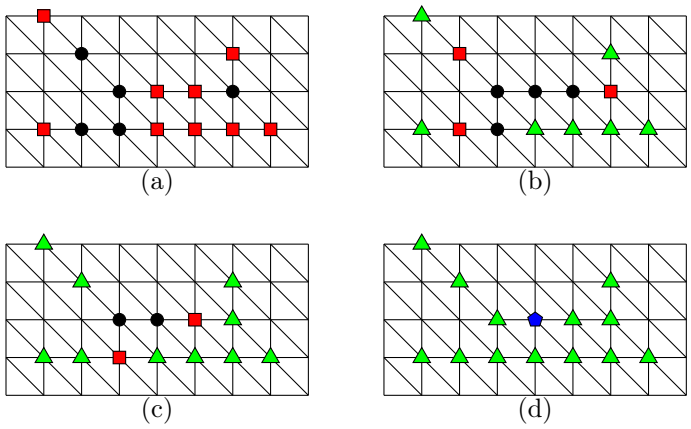


Figure 6: An example of the execution of S-contraction algorithm after one round (Figure 6.a), two rounds (Figure 6.b), three rounds (Figure 6.c) and after four rounds (Figure 6.d; circle: non S-contractible particle; square: S-contractible particle; triangle: particle in state **N**; pentagon: particle in state **L**; S being the set of particles in state **C**).

1 is illustrated by Figure 6. Note that, depending on the order in which the local computations occur, the result of the execution of the algorithm could be different. For example, between the configuration of Figure 6.c and that of Figure 6.d, we suppose that the local computations occur in this order: first a local computation occurs for the bottom left particle, second it occurs for the upper left particle, third it occurs for the upper right particle and fourth it occurs for the last particle (we only consider the particles which are in state **C**).

Algorithm 1 The S-contraction algorithm for a particle p and S the set of particles in state **C**.

Case 1: State **C**.
if the particle is S-contractible **then**
  if the particle has no neighbor in S **then**
    set the state to **L**.
  else
    set the state to **N**.
  end if
else
  stay in state **C**.
end if
Case 2: States **L** or **N**.
Perform no further actions.

Theorem 1. *Let S be the set of particles in state **C** and P be the particle graph on G. If P is hole-free, then at the end of the execution of the S-contraction algorithm, there will be exactly one particle in the state L.*

In Appendix F, the proof of Theorem 1 is given. Also, a bound on the complexity of the S-contraction algorithm is given. In Appendix H, it is explained how to combine the S-contraction algorithm with a general leader election algorithm.

4 Assigning k-local identifiers to particles

In this section, we combine the results from Section 3 and Appendix I in order to correctly compute a k-local identifier. In a first subsection, we describe a way to create a spanning tree of particles and a way to change the ports numbering of the different particles. In a second subsection, we describe how to compute k-local identifiers based on the coloring functions from Appendix I.

We suppose that Algorithms 2 and 3 are preceded by a leader election algorithm (which could be Algorithm 1). Then it follows that there is a single particle in a specific state (the leader) and all the remaining particles are in the same state (non elected).

4.1 Re-organizing the particles

By $N_G^+(u)$ we denote the set of port numbers which can communicate with particles occupying vertices from $N_G(u)$. When there is a leader, we can easily compute a spanning tree using a distributed algorithm (see Appendix G). Now suppose that for each particle p, we have two set of ports parent(p) and child(p) which contains the port numbers of the particles in communication with its parent and with its children, respectively, in the spanning tree. In this way, the required memory in order to store where are the children and the parent of the particle in a spanning tree is constant (since the maximum degree is bounded in the considered grids).

In our proposed Algorithm 2, the goal is to change the way the port are numbered in order that every particle has its ports numbered by the same number going in the same cardinal direction in the different grids. This algorithm does not work if we do not have a leader among the different particles. The function r_G used in Algorithm 2 is defined, depending the choice of G, as follows: $r_{\mathbb{S}}(i) = (i+2) \pmod 4$, $r_{\mathcal{T}}(i) = (i+3) \pmod 6$ and $r_{\mathcal{K}}(i) = (i+3) \pmod 6$.

Algorithm 2 The port renumbering algorithm for a particle p.

Case 1: State **L**.
for each port a from child(p) send a message m_a, containing a, through port a.
Case 2: State **N**.
if the particle receives the message m_b, containing b, through the port a **then**
change the port number a to $r_G(b)$ and changes the port numbers of the other ports following the clockwise order;
update both parent(p) and child(p).
end if

The idea behind Algorithm 2 is to reproduce, in each particle, the way the ports are numbered in the leader particle. To achieve this goal, each particle p receives a message from its parent containing the port number of the parent connected to p and p renumbers its own ports in order that its port numbers are coherent with the sent number. Figure 7.a and Figure 7.b illustrate the port numbers of particles before and after the execution of Algorithm 2.

4.2 The k-local identifiers

Now, we aim to give to each particle a variable id, called its k-local identifier, such that every two particles p_1 and p_2 with the same identifier satisfy $d_G(s(p_1), s(p_2)) > k$. If we suppose that the particles have not a memory of at least $\log_2(n)$ bits, for $n = |P|$, then it is not possible to record a unique variable for each particle. However, it is possible to have a k-local identifier in the three considered grids only using at most $\log_2((k+1)^2)$ bits where k is a parameter given by the user. Our proposed Algorithm 3 presents an optimal way (in term of memory) to compute k-local identifiers. We suppose that the port renumbering algorithm (Algorithm 2) has been done before executing Algorithm 3.

Algorithm 3 consists in assigning a variable which corresponds to a color in a coloring of the k^{th} power on the grid. More precisely, the function f_G^k consists in assigning a color depending the Cartesian coordinate of the vertices. Since the colors are given following a pattern, the

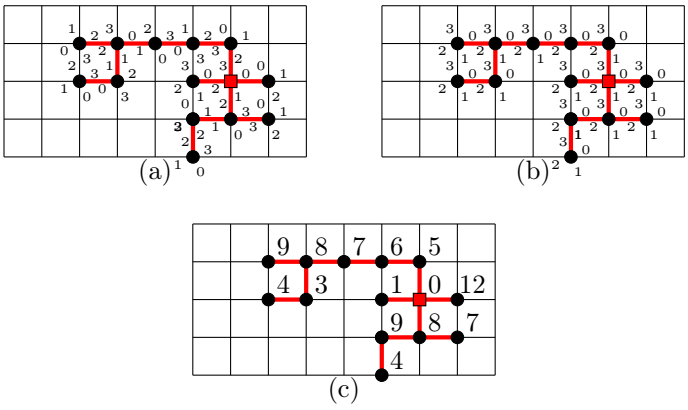


Figure 7: One spanning tree of particles, a possible numbering of the ports of the particles before (Figure 7.a) and after the execution of Algorithm 2 (Figure 7.b) and the 4-identifier obtained by executing Algorithm 3 (Figure 7.c) in the square grid (square: leader; thick line: edge of the spanning tree; small number: port number of a particle; big number: 4-identifier of a particle).

Algorithm 3 The k-local identifier algorithm for a particle p.

Case 1: State **L**.
set $i = 0$, $j = 0$, $id = 0$;
send i and j through each port from child(p).
Case 2: State **N**.
if the particle receives the integers i' and j' through the port a **then**
set $i = I_G^k(i', a)$, $j = I_G^k(j', a)$, $id = f_G^k(i, j)$;
send i and j through each port from child(p).
end if

Cartesian coordinate can be stored relatively to the size of the patterns. In Algorithm 3, the leader affects to itself the color 0 and following the direction where the messages are transmitted, the particles reproduce the coloring patterns given in Appendix I. The functions f_G^k and I_G^k, J_G^k, used in Algorithm 3 are defined, depending on the choice of G, as follows: $f_{\mathcal{S}}^k(i,j) = (i + kj) \pmod{m_k}$, $f_{\mathcal{K}}^k(i,j) = i \pmod{k+1} + (k+1)j \pmod{k+1}$ and $f_{\mathcal{T}}^k(i,j) = (i \pmod{3(k+1)/2} + j(3(k+1)/2) + \lfloor 2j/(k+1) \rfloor (k+1)/2)) \pmod{m'_k}$ if k is odd or $f_{\mathcal{T}}^k(i,j) = (i + (3k/2+1)j) \pmod{m'_k}$ otherwise; $I_G^k(i,a) = i$ if $(a = 1;3 \wedge G \cong \mathcal{S}) \vee (a = 1;4 \wedge G \cong \mathcal{T}) \vee (a = 2;6 \wedge G \cong \mathcal{K})$, $I_{\mathcal{S}}^k(i,a) = i + 1 \pmod{\lceil (k+1)^2/2 \rceil}$ if $a = 0$, $I_{\mathcal{S}}^k(i,a) = i - 1 \pmod{\lceil (k+1)^2/2 \rceil}$ if $a = 2$, $I_{\mathcal{T}}^k(i,a) = i + 1 \pmod{\lceil 3(k+1)^2/4 \rceil}$ if $a = 0;5$, $I_{\mathcal{T}}^k(i,a) = i - 1 \pmod{\lceil 3(k+1)^2/4 \rceil}$ if $a = 2;3$, $I_{\mathcal{K}}^k(i,a) = i+1 \pmod{k+1}$ if $a = 0;1;7$ and $I_{\mathcal{K}}^k(i,a) = i-1 \pmod{k+1}$ if $a = 3;4;5$; $J_G^k(j,a) = i$ if $(a = 0;2 \wedge G \cong \mathcal{S}) \vee (a = 0;6 \wedge G \cong \mathcal{T}) \vee (a = 0;4 \wedge G \cong \mathcal{K})$, $J_{\mathcal{S}}^k(j,a) = j+1 \pmod{\lceil (k+1)^2/2 \rceil}$ if $a = 1$, $J_{\mathcal{S}}^k(j,a) = i - 1 \pmod{\lceil (k+1)^2/2 \rceil}$ if $a = 3$, $J_{\mathcal{T}}^k(j,a) = i + 1 \pmod{\lceil 3(k+1)^2/4 \rceil}$ if $a = 1;2$, $J_{\mathcal{T}}^k(j,a) = i-1 \pmod{\lceil 3(k+1)^2/4 \rceil}$ if $a = 4;5$, $J_{\mathcal{K}}^k(j,a) = i+1 \pmod{k+1}$ if $a = 1;2;3$ and $J_{\mathcal{K}}^k(j,a) = i - 1 \pmod{k+1}$ if $a = 5;6;7$. Note that the functions I_G^k and J_G^k are used to determine the Cartesian coordinate of a particle using the Cartesian coordinate of a neighbor and the port number of this neighbor.

Since the values of $f_G^k(i,j)$ is bounded by $3(k+1)^2/4$, if G is isomorphic to one of the three grids, the size of the messages will not exceed $\log_2(3(k+1)^2/4)$. As for Algorithm 2, the number of sent messages is $|V(P)| - 1$. Figure 7.c illustrates the obtained 4-identifiers after the execution of Algorithm 3.

Since the particles can move during the execution of an algorithm, the k-local identifiers may become not valid anymore (i.e., there may be two particles p_1 and p_2 with the same k-local identifier and with $d_G(s(p_1), s(p_2)) \leq k$) if the structure of the particle graph P on G changes. It

is possible to keep a valid k-local identifier in case a particle moves in a direction of a port a by setting $id = f_G^k(I_G^k(i, r_G(a)), J_G^k(j, r_G(a)))$ as the new k-local identifier. It corresponds to update the variable id which corresponds to a color in a coloring of the k^{th} power on the grid in function of the new position of the particle. Also, in the case a particle do ℓ movements, by storing the successive directions of movement of the particle during these ℓ movements, it is also possible to update the value of the k-local identifier in order that it remains valid.

Note that for both Algorithms 2 and 3 finish after at most h rounds, h being the height of the spanning tree. Also the number of sent messages in both Algorithms 2 and 3 is $|V(G)| - 1$ (the number of edges in a spanning tree).

5 Conclusion

In this paper, we have presented a new leader election algorithm based on local computation. We have also presented an algorithm which affects a different variable for every two particles p_1 and p_2 at distance at most k. All the presented algorithms only require a $O(1)$-space memory. This complexity makes it possible to use our algorithms for programmable matter. Moreover, in case of movements of particles, there is no need of communication in order to update the k-local identifiers.

As future work, it would be interesting to determine a more general deterministic leader election algorithm in our algorithmic framework that can take into account fault tolerance. Also, it would be interesting to extend the presented results to 3D grids. Another interesting question could be to use our results to clustering the set of particles in several sets which induce subgraphs of small diameter.

Acknowledgments

This work was supported by the French "Investissements d'Avenir" program, project ISITE-BFC (contract ANR-15-IDEX-03).

References

[1] H. Attiya, J. Welch, Distributed computing: fundamentals, simulations and advance topics, *John Wiley and Sons*, 2004.

[2] M. Bauderon, Y. Métivier, M. Mosbah, A. Sellami, Graph relabelling systems: a tool for encoding, proving, studying and visualizing distributed algorithms, *Electronic Notes in Theoretical Computer Science* **51**, 93–107, 2001.

[3] J. Bourgeois, S. Copen Goldstein, Distributed intelligent MEMS: progresses and perspectives, *IEEE Systems Journal* **9** (3), 1057–1068, 2015.

[4] Z. J. Butler, K. Kotay, D. Rus, K. Tomita, Generic decentralized control for lattice-based self-reconfigurable robots, *International Journal of Robotics Research*, **23**(9), 919–937, 2004.

[5] J. J. Daymude, R. Gmyr, A. W. Richa, C. Scheideler, T. Strothmann, Improved leader election for self-organizing programmable matter, *ALGOSENSORS 2017: Algorithms for Sensor Systems*, 127–140, 2017.

[6] Z. Derakhshandeh, S. Dolev, R. Gmyr, A. W. Richa, C. Scheideler, T. Strothmann, Brief announcement: amoebot - a new model for programmable matter, *SPAA 2014*, 220–222, 2014.

[7] Z. Derakhshandeh, R. Gmyr, A. W. Richa, C. Scheideler, T. Strothmann, An algorithmic framework for shape formation problems in self-organizing particle systems, *NANOCOM 2015*, **21**, 1–21, 2015.

[8] Z. Derakhshandeh, R. Gmyr, T. Strothmann, R. A. Bazzi, A. W. Richa, C. Scheideler, Leader election and shape formation with self-organizing programmable matter, *DNA 2015*, 117–132, 2015.

[9] Z. Derakhshandeh, R. Gmyr, A. W. Richa, C. Scheideler, T. Strothmann, Universal shape formation for programmable Matter, *SPAA 2016*, 289–299, 2016.

[10] Z. Derakhshandeh, R. Gmyr, A. Porter, A. W. Richa, C. Scheideler, T. Strothmann, On the runtime of universal coating for Programmable Matter, *DNA 2016*, 148–164, 2016.

[11] Z. Derakhshandeh, R. Gmyr, A. W. Richa, C. Scheideler, T. Strothmann, Universal coating for programmable matter, *Theor. Comput. Sci.* **671**, 56–68, 2017.

[12] G. Fertin, E. Godard, A. Raspaud, Acyclic and k-distance coloring of the grid, *Inform. Process. Lett.* **87** (1), 51–58, 2003.

[13] G. A. Di Luna, P. Flocchini, N. Santoro, G. Viglietta, Y. Yamauchi, Shape Formation by Programmable Particles, *OPODIS 2017*.

[14] H. Lakhlef, H. Mabed, J. Bourgeois, Optimization of the logical topology for mobile MEMS networks, *J. Network and Computer Applications* **42**, 163–177, 2014.

[15] A. Itai, M. Rodeh, Symmetry breaking in distributed networks, *Annual Symposium on Foundations of Computer Science*, 150–158, 1981.

[16] P. Jacko, S. Jendrol, Distance coloring of the hexagonal lattice, *Discuss. Math. Graph Theory* **25**, 151–166, 2005.

[17] F. Kramer, H. Kramer, A survey on the distance-colouring of graphs, *Discrete Math.* **308**, 422–426, 2008.

[18] A. Naz, B. Piranda, J. Bourgeois, S. Copen Goldstein, A distributed self-reconfiguration algorithm for cylindrical lattice-based modular robots, *NCA*, 254–263, 2016.

[19] A. Mazurkiewicz, Distributed enumeration, *Inform. Process. Letters* **61**, 233–239, 1997.

[20] B. Piranda, J. Bourgeois, Geometrical Study of a Quasi-spherical Module for Building Programmable Matter, *Distributed Autonomous Robotic Systems*, 347–400, 2018.

[21] P. Rosenstiehl, J. -R. Fiksel, A. Holliger, Intelligent graphs, *Graph theory and computing*, 219–265, 1972.

[22] A. Sevcikova, Distant chromatic number of the planar graphs, *Manuscript*, Safarik University, 2001.

[23] T. Tucci, B. Piranda, J. Bourgeois, Efficient scene encoding for programmable matter self-reconfiguration algorithms, *SAC*, 256–261, 2017.

[24] M. Yim, W-M. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins, G. S. Chirikjian, Modular self-reconfigurable robot systems, *IEEE Robotics Automation Magazine*, **14** (1), 43–52, 2007.

F Proof of Theorem 1 and bound on the complexity of the S-contraction algorithm

The three lemmas presented in this appendix are used in order to prove Theorem 1.

In the following lemma, we describe how to determine, in the context of programmable matter, if a particles is S-contractible or not.

Lemma 1. *Let G be an infinite grid among $\mathcal{S}$, $\mathcal{T}$ and $\mathcal{K}$ and let $S \subseteq V(P)$, for P the particle graph on G and S the set of vertices occupied by all the particles in the same fixed state. One round is sufficient in order that every particle determine if it is S-contractible or not if G is isomorphic to $\mathcal{S}$. Otherwise, if G is isomorphic to $\mathcal{T}$ or $\mathcal{K}$, no round is necessary.*

Proof. Let $N^+(p)$ be the set of port labels on which p can communicate with particles from its neighborhood. In order to verify that $M_G(p) \cap S$ is connected in the triangular or king grids, it suffices to verify that $N_G^+(p) \cap S$ forms an interval of consecutive integers (by considering that 0 and $\Delta(G)-1$ are consecutive). For example, $\{0, 4, 5\}$ contains successive integers in the triangular grid but that is not the case for $\{0, 2, 5\}$. Such verification in the triangular and king grids can be done during any local computation. Figure 5 illustrates three possible cases that could happen for a particle in the triangular grid. On the left part of Figure 5, the particle does not satisfy Property I) but satisfies Property II). On the middle part of Figure 5, the particle satisfies Property I) and does not satisfy Property II). Finally, on the right part of Figure 5, the particle satisfies both Properties I) and II).

In the square grid, in order to test if a particle p is such that $M_G(p) \cap S$ is connected, it requires to receive $N_G^+(p') \cap S$, from the particle p' in the neighborhood of p and afterward to test if $N_G^+(p)$ only contains consecutive integers (by considering that 0 and 3 are consecutive) and then to verify, for any two successive particles p' and p'' from the neighborhood, that the vertex which corresponds to the corner adjacent to both p' and p'' is occupied by a particle.

If G is among $\mathcal{T}$ and $\mathcal{K}$, then no round is required to know if a particle is in S or not (since a particle know the state of its neighbors). If G is isomorphic to $\mathcal{S}$, then, in one round, which consists in sending the values of $N^+(p) \cap S$ to the adjacent particles, every particle knows if it is S-contractible or not. □ □

The following lemma is be useful in order to prove that our leader election algorithm works correctly.

Lemma 2. *Let G be an infinite grid among $\mathcal{S}$, $\mathcal{T}$ and $\mathcal{K}$ and let $S \subseteq V(P)$, for P the particle graph on G. Let p be an S-contractible particle. If S is connected and hole-free, then $S - \{s(p)\}$ is connected and hole-free.*

Proof. First, note that in all three grids, the fact that $|N_G(p) \cap S| < |N_G(p)|$ implies that there is a vertex v in $N_G(p) \setminus S$. By contradiction, suppose we create a hole in $G[S]$ by removing the vertex $s(p)$ from S. This implies, since $G[M_G(p) \cap S]$ is connected, that v was already in a hole from $G[S]$. Second, since $G[M_G(p) \cap S]$ is connected, we are sure that the subgraph $G[S \setminus \{s(p)\}]$ is connected. □ □

To ensure that our leader election algorithm works correctly, it remains to prove that there always exists an S-contractible particle. That is what we do in the following Lemma.

Lemma 3. *Let $S \subseteq V(P)$, for P the particle graph on G. If $G[S]$ is hole-free, then there always exists an S-contractible particle in S.*

Proof. Note that there exists a particle on the border of $G[S]$ since S is finite. Let A be the set of particles on the border of $G[S]$. For any particle p, the fact that there are at least two connected components B_1 and B_2 in $G[M_G(p) \cap S]$ implies that there is no path in $G[S \setminus \{s(p)\}]$ between any vertex of B_1 and a vertex of B_2, since it would imply the existence of a hole in $G[S]$ containing a vertex from $M_G(p) \setminus S$. Therefore, if $p \in A$ and if p is not an S-contractible particle, then p is an articulation of $G[S]$.

Now suppose, by contradiction, that there is no S-contractible particle in S. By the previous remark, the graph $G[A]$ is connected and all particles of A are articulations of $G[S]$. However, a finite graph containing a cycle contains vertices which are not articulation of $G[S]$. Thus, $G[A]$ contains no cycle ($G[A]$ is a forest). However, by definition, the leaves (the vertices of degree 1) are S-contractible. Thus, we obtain a contradiction with the fact that there is no S-contractible particle in S. □ □

In the case P is hole-free, note that by Lemma 2 and Lemma 3 there is always a particle which is both on the border of $G[S]$ and not an articulation of $G[S]$.

Proof of Theorem 1. Note that before the execution of the algorithm, the set S is the set $V(P)$. Since P is hole-free and connected and by Lemma 2, S remains connected and hole-free during the execution of the algorithm. By Lemma 3, there is always a particle in S which is S-contractible (every particle on the border which is not an articulation is S-contractible). Thus, for every round, the number of particles in state **C** strictly decreases. Since $|V(P)|$ is finite, we are sure that at some point, S will only contain one vertex. If at some point, S contains one vertex, there will be at least one elected leader.

Finally, note that the fact that there are two elected leaders contradicts the fact that S remains connected during the execution of the algorithm. □ □

Let G' be a subgraph of G such that G' is hole-free, the *radius* of G', denoted by $r(G')$, is given by $r(G') = \min_{u \in V(G')} \{\max_{v \in A}(d_{G'}(u,v))\}$, for A the set of the particles on the border of G'. Moreover, let $h(T)$ be the *height* of a tree T, i.e., $h(T) = \min_{u \in V(T)} \{\max_{v \in V(T),\ |N_T(v)|=1} (d_T(u,v))\}$ and let $mtree(G')$ be the maximum height among all induced subgraphs of G' which are trees, i.e., among the set $\{G'[B] |\ B \subseteq V(G'),\ G'[B]$ is a tree $\}$.

In the following Proposition, we give a bound on the required number of rounds for the termination of Algorithm 1.

Proposition 1. *Let S be the set of particles in state **C** and P be the particle graph on G. Moreover, let $b_G = r(P) + mtree(P) + 1$ if G is isomorphic to $\mathcal{T}$ or $\mathcal{K}$ or $b_G = 2(r(P) + mtree(P)) + 2$ if G is isomorphic to $\mathcal{S}$. If P is hole-free, then after $b_G(P)$ rounds of the S-contraction algorithm on P, one particle will be the leader.*

Proof. First, suppose G is isomorphic to $\mathcal{T}$ or $\mathcal{K}$. Let S_t be the set of particles in state **C** after the first t rounds. Note that after $r(P)+1$ rounds we are sure that every remaining particle u satisfies $|N_G(u) \cap S_{r(S)+1}| < N_G(u)$. This is due to the fact that each particle u on the border of S_i, for $i \geq 0$, is not in S_{i+1} if $|N_G(u) \cap S_{r(S)+1}| < N_G(u)$. Thus, by Lemma 2, $G[S_{r(P)+1}]$ is either a tree or empty. By definition, we have $h(G[S_{r(P)+1}]) \leq mtree(P)$. Note that in the case $G[S_t]$ is not a trivial tree (a tree containing only one vertex), we have $h(G[S_{t+1}]) = h(G[S_t]) - 1$, for $t \geq r(P)+1$. Therefore, we obtain that S_t is empty if $t \geq r(P) + mtree(P) + 1$.

Second, suppose G is isomorphic to $\mathcal{S}$. Note that, by Lemma 1, one round is sufficient in order that every particle determines if it is S-contractible. Consequently, it is easy to observe that the required number of rounds in order that the S-contraction algorithm finishes for $\mathcal{S}$ is bounded by two times the required number of rounds in order that the S-contraction algorithm finishes for $\mathcal{T}$ or $\mathcal{K}$. □ □

G An example of algorithm in order to construct a spanning tree

Our proposed algorithm (Algorithm 4) for constructing a spanning tree consists in setting the particle in state **L** as the root and, afterward, constructing a spanning tree using a classical distributed spanning tree algorithm.

H Combining the S-contraction algorithm with a general leader election algorithm

Daymude et al. [5] introduced a leader election algorithm that works on every configuration of P. In this appendix we present a way to reduce the required number of rounds in order that this algorithm finishes its execution (by using the S-contraction algorithm). In the remaining part of this appendix, the leader election algorithm from [5] will be called the *general leader election*

Algorithm 4 A spanning-tree algorithm for a particle p.

Case 1: State **L** (leader).
set child$(p) = N_G^+(p)$;
send a message m (which only contains the bit 0) through each port from child(p).
Case 2: State **N** (not elected).
if the particle receives the message m through the port a **then**
  if the particle has never received the message m before **then**
    set parent$(p) = a$;
    set child$(p) = N_G^+(p) \setminus \{a_1, \ldots, a_\ell\}$, where a_1, ..., a_ℓ are ports on which p has received the message m;
    send the message m through each port from child(p).
  end if
end if

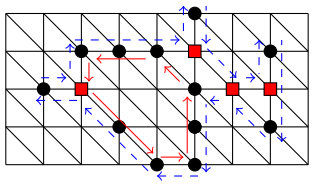


Figure 8: The way how the messages are re-transmitted in the algorithm of Daymude et al. [5] (square: articulation; dashed arrow: message transmitted along the border; simple arrow: message transmitted along the hole).

algorithm (to the best of our knowledge, it is the only leader election algorithm for programmable matter working on every configuration).

In order to simplify the presentation of the results, we only discuss the results for the triangular grid. However, by modifying the algorithms it is possible to make it work for the square and king grids also. We begin this appendix by describing how the general leader election algorithm works. This description will help the reader to convince itself that, in a lot of cases, combining the S-contraction algorithm with a general leader election algorithm could be a good idea.

For particle p and a port a of p connected to another particle, we denote by $n(a)$ the port number of the first port of p connected to a particle after a in the clockwise order. The general leader election algorithm uses the fact that it is possible to send a message around a boundary. Sending a message around a boundary consists in sending and re-transmitting the message in the following way: for a particle p, if a message is received from the port a, then the particle re-transmits the message to the particle connected to p by the port $n(a)$ of p. Figure 8 represents how the messages are transmitted in this case.

For each hole H of the particle graph P on G, we denote by $b(H)$, the set of particles which are adjacent with a vertex of H. Also, when a particle is adjacent to vertices of different holes or when a particle belongs to the border of P, it is possible to decompose particles in agents, each agent corresponding to a different hole or to the border. Thus, an agent will be either adjacent to vertices of at most one hole or belong to the border but never both.

The general leader election algorithm can be summarized as the succession of four phases. A first phase consists in removing the candidacy of each particle having six neighbors. A second phase consists, for each hole H of P, to remove the candidacy of some agents of $b(H)$ using a randomized procedure. Simultaneously, the same process is done for agents in the border of P. A third phase consists in verifying if there is only one candidate in $b(H)$, for each hole H of P and only one candidate in the border of P. They calculate the relative positions of the candidates in order to do such verification. Finally, in the last phase, they verify if the remaining candidates agents are in $b(H)$ or in border of P. The leader will be the candidate particle of the border of P. We can

verify if an agent is in $b(H)$ by sending a message around the boundary and verifying when the message comes back to the initial particle if this message has been re-transmitted in the clockwise direction or not. As Figure 8 illustrates, the messages re-transmitted through the boundary of a hole are re-transmitted in the counterclockwise direction and the messages re-transmitted through the border of P are re-transmitted in the clockwise direction. In all these phases, the messages are re-transmitted around a boundary.

The required number of rounds in order that the general leader election algorithm finishes its execution is $O(\ell)$, where ℓ is the number of particles in the border of P.

We do the following remark about the S-contraction algorithm that comes from the fact that for any S-contractible particle p of S, if $G[S]$ is connected, then $G[S \setminus \{s(p)\}]$ is also connected.

Remark 1. *If the particle graph P has a hole then, after the execution of the S-contraction algorithm on the particle graph P on G there will remain particles in state **C**. Also, the graph induced by the particles in state **C** is connected.*

In particular when P has one hole, the remaining particles in state **C** will form a ring in the triangular grid. Thus, it is possible to run the S-contraction algorithm and, afterward, execute the general leader election algorithm on the remaining particle in state **C**. Let S_c be the set of particles in state **C** after the S-contraction algorithm on the particle graph P on G. Depending on the structure of P, it could happen that the number of particles in the border $\mathscr{T}[S_c]$ is smaller than in the border of P and that it speeds the execution of the general leader election algorithm. For example, that is always the case when P has at most one hole.

I Coloring the k^{th} power of graph

The k^{th} power of a graph G is the graph on the same vertex set than G and with edges connecting every two vertices u and v satisfying $d_G(u, v) \leq k$. Note that there is a correlation between this definition and the k^{th} power of the adjacency matrix of G (the adjacency matrix of the k^{th} power of G is easily obtained from this matrix). Our goal in this appendix is to determine an optimal coloring of the k^{th} power of the square, triangular and king grids. We use these colorings in order to propose a distributed algorithm (supposing we have a leader) in order to assign k-local identifiers to the particles (see Section 4). A coloring of the k^{th} power of a grid corresponds to assign a value to each vertex of the graph such that every two vertices with the same assigned value are at distance at least $k + 1$. An example of coloring of the k^{th} power of the square grid is represented by Figures 9 and 10. In Figure 9, it is easy to notice that every two vertices with color 0 (or any other color) are at distance at least 4.

More formally, a *k-coloring* of a graph G is a map c from $V(G)$ to $\{0, 1, \ldots, k-1\}$ which satisfies $c(u) \neq c(v)$ for every $uv \in E(G)$. The *chromatic number* $\chi(G)$ of G, is the smallest integer k such that there exists a k-coloring of G. The *k^{th} power* G^k of a graph G is the graph obtained from G by adding an edge between every two vertices satisfying $d_G(u, v) \leq k$. More details about the coloring of the k^{th} power of graphs can be found in the survey from Kramer and Kramer [17]. The results presented in this appendix are inspired by the previous works [12, 16, 22] about the coloring of the k^{th} power of the grids.

Appendix I.A Coloring the k^{th} power of square grids

We give the following result from Fertin et al. [12].

Theorem 2 ([12])**.** *For any $k \geq 1$, $\chi(\mathcal{S}^k) = \lceil (k+1)^2/2 \rceil$.*

Let $m_k = \lceil (k+1)^2/2 \rceil$. In their paper, Fertin et al. define an optimal coloring c of the k^{th} power of the square grid as follows: $c((i, j)) = (i + kj) \pmod{m_k}$. In Figures 9 and 10, we represent patterns for coloring the $3th$ and $4th$ powers of the square grid. These patterns have been obtained using the coloring from [12]. Note that since there is a pattern, a vertex (i, j) can determine its color only knowing $i \pmod{m_k}$ and $j \pmod{m_k}$. We recall the definition of

0	1	2	3	4	5	6	7
3	4	5	6	7	0	1	2
6	7	0	1	2	3	4	5
1	2	3	4	5	6	7	0
4	5	6	7	0	1	2	3
7	0	1	2	3	4	5	6
2	3	4	5	6	7	0	1
5	6	7	0	1	2	3	4

Figure 9: A pattern for coloring the 3^{th} power of the square grid.

the following function $f^k_{\mathcal{S}}(i,j) = (i + kj) \pmod{m_k}$. Note that $f^k_{\mathcal{S}}(i,j) = f^k_{\mathcal{S}}(i',j')$, in the case $i \equiv i' \pmod{m_k}$ and $j \equiv j' \pmod{m_k}$ This function is used in order to assign k-local identifiers to particles.

Appendix I.B Coloring the k^{th} power of triangular grids

The chromatic number of the k^{th} power of the triangular grid has been determined by Sevcikova [22].

Theorem 3 ([22]). *For any $k \geq 1$, $\chi(\mathcal{T}^k) = \lceil 3(k+1)^2/4 \rceil$.*

Let $m'_k = \lceil 3(k+1)^2/4 \rceil$. We recall the definition of the following function:

$$f^k_{\mathcal{T}}(i,j) = \begin{cases} (i \pmod{3(k+1)/2} + j(3(k+1)/2) + \\ \lfloor 2j/(k+1) \rfloor (k+1)/2)) \pmod{m'_k} & \text{if } k \text{ is odd;} \\ (i + (3k/2+1)j) \pmod{m'_k} & \text{otherwise.} \end{cases}$$

Note that $f^k_{\mathcal{T}}(i,j) = f^k_{\mathcal{T}}(i',j')$, in the case $i \equiv i' \pmod{m'_k}$ and $j \equiv j' \pmod{m'_k}$. This function is used in order to assign k-local identifiers to particles.

Appendix I.C Coloring the k^{th} power of king grid

To our knowledge, the chromatic number of the king grid has not been determined yet. However, in contrast with the triangular grid, the chromatic number of the k^{th} power of the king grid is easy to determine. In this subsection, we determine the exact value of the chromatic number of the k^{th} power of the king grid.

Theorem 4. *We have $\chi(\mathcal{K}^k) = (k+1)^2$.*

Proof. Let $\mathcal{K}_k$ be the subgraph of $\mathcal{K}$ induced by the vertices $\{(i,j) \in V(\mathcal{K}) |\ 0 \leq i \leq k,\ 0 \leq j \leq k\}$. Note that $diam(\mathcal{K}_k) = k$ and that $|V(\mathcal{K}_k)| = (k+1)^2$. Thus, since each vertex of $\mathcal{K}_k$ must be colored differently in a coloring of the k^{th} power of the king grid, we obtain that $\chi(\mathcal{K}^k) \geq (k+1)^2$. We define the coloring function $c((i,j)) = i \pmod{k+1} + (k+1)j \pmod{k+1}$. Note that we have $d_{\mathcal{K}}(u,v) \geq k+1$, for every two vertices u and v with the same color in $\mathcal{K}$. Therefore, we obtain that $\chi(\mathcal{K}^k) = (k+1)^2$. □ □

Note that since there is a pattern, a vertex (i,j) can determine its color only knowing $i \pmod{(k+1)}$ and $j \pmod{(k+1)}$. We recall the definition of the following function $f^k_{\mathcal{K}}(i,j) = i \pmod{k+1} + (k+1)j \pmod{k+1}$. Note that $f^k_{\mathcal{K}}(i,j) = f^k_{\mathcal{K}}(i',j')$, in the case $i \equiv i' \pmod{k+1}$ and $j \equiv j' \pmod{k+1}$. This function is used in order to assign k-local identifiers to particles.

0	1	2	3	4	5	6	7	8	9	10	11	12
5	6	7	8	9	10	11	12	0	1	2	3	4
10	11	12	0	1	2	3	4	5	6	7	8	9
2	3	4	5	6	7	8	9	10	11	12	0	1
7	8	9	10	11	12	0	1	2	3	4	5	6
12	0	1	2	3	4	5	6	7	8	9	10	11
4	5	6	7	8	9	10	11	12	0	1	2	3
9	10	11	12	0	1	2	3	4	5	6	7	8
1	2	3	4	5	6	7	8	9	10	11	12	0
6	7	8	9	10	11	12	0	1	2	3	4	5
11	12	0	1	2	3	4	5	6	7	8	9	10
3	4	5	6	7	8	9	10	11	12	0	1	2
8	9	10	11	12	0	1	2	3	4	5	6	7

Figure 10: A pattern for coloring the 4^{th} power of the square grid.