IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 3, 2020, accepted November 9, 2020, date of publication November 16, 2020,
date of current version November 30, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3038174

Asynchronous Silent Programmable Matter Achieves Leader Election and Compaction

GIANLORENZO D'ANGELO[1], MATTIA D'EMIDIO[2], SHANTANU DAS[3], ALFREDO NAVARRA[4], AND GIUSEPPE PRENCIPE[5]

[1]Gran Sasso Science Institute, 67100 L'Aquila, Italy
[2]Department of Information Engineering, Computer Science and Mathematics, University of L'Aquila, 67100 L'Aquila, Italy
[3]Aix-Marseille Université, CNRS, Université de Toulon, LIS, 13288 Marseille, France
[4]Department of Mathematics and Computer Science, University of Perugia, 06123 Perugia, Italy
[5]Department of Computer Science, University of Pisa, 56126 Pisa, Italy

Corresponding author: Giuseppe Prencipe (giuseppe.prencipe@unipi.it)

This work was supported in part by Ministero dell'Istruzione, Università e Ricerca (MIUR) PRIN 2017 Project ALGADIMAR "Algorithms, Games, and Digital Markets", in part by Gruppo Nazionale per il Calcolo Scientifico dell'Istituto Nazionale di Alta Matematica (GNCS-INdAM), in part by the Università di Pisa, Italy, under Grant Progetto PRA_2018_43, and in part by the Project "ASSIOMI–Algorithms, Systems and Devices for Machine Monitoring and Diagnosing in Smart Factories (Algoritmi Sistemi e diSpositivI per mOnitoraggio e diagnostica di Macchine per le fabbriche Intelligenti).

ABSTRACT We study models and algorithms for *Programmable Matter* (PM), that is matter with the ability to change its physical properties (e.g., shape or optical properties) in a programmable fashion. PM can be implemented by assembling a system of weak self-organizing computational elements, called *particles*, that can be programmed via distributed algorithms to collectively achieve some global task. Recent advances in the production of nanotechnologies have rendered such systems increasingly possible in practice, thus triggering research interests from many areas of computer science. The most established models for PM assume that particles: are modeled as finite state automata; are all identical, executing the same algorithm based on local observation of the surroundings; live and operate in the cells of a hexagonal grid; can move from one cell to another by repeatedly alternating between a *contracted* state (a particle occupies one cell) and an *expanded* state (a particle occupies two neighboring cells). Given these elementary features, it is rather hard to design distributed algorithms even for basic tasks and, in fact, all existing solutions to solve fundamental problems via PM have resorted to endowing PM systems with various capabilities to overcome such hardness, thus assuming quite unrealistic features. In this paper, we move toward more realistic computational models for PM. Specifically, we first introduce SILBOT, a new modeling approach that relaxes several assumptions used in previous ones. Second, we present a distributed algorithm to solve, in the SILBOT model, a foundational primitive for PM, namely *Leader Election*. This algorithm works in $O(n)$ rounds for all initial configurations of n particles that are both *connected* (i.e. particles induce a connected graph) and *compact* (i.e. without *holes*, that is no empty cells surrounded by particles occur). As usual in asynchronous contexts, a *round* is intended as the time within which all particles have been activated at least once. Third, we show that, if the initial configuration admits holes, it is impossible to achieve leader election while preserving connectivity. Finally, by slightly empowering the robots, we design an algorithm to handle initial configurations admitting holes that in $O(n^2)$ rounds solves the leader election problem while obtaining also compaction.

INDEX TERMS Programmable matter, swarm robotics, self-organizing systems, leader election, compaction, finite automata, distributed algorithms.

I. INTRODUCTION

Matter having the ability to change its physical properties (e.g., shape, optical properties, etc.) in a programmable fashion has been recently the subject of many studies in many areas of computer science, including robotics and distributed computing.

The term *Programmable Matter* (PM, shortly) was first coined in a seminal work by Toffoli *et al.* [37] and, since the beginning, it has been used to denote systems of weak and small computational elements, called *particles*, that can be programmed via distributed algorithms to

The associate editor coordinating the review of this manuscript and approving it for publication was Khalid Aamir.

collectively achieve some global task. Particles are weak in the sense that they are able to perform, in some self-organized way, very simple actions only, such as establishing and releasing bonds or moving in geometrically constrained environments.

Initially, the interest by the scientific community was mostly theoretical, as in the early 90s the technology for building computational devices at micro/nanoscale was still in a nascent state. Nowadays, instead, such devices have been rendered increasingly possible in practice thanks to the advancements in the production processes of nanounits that integrate computing, sensing, actuation, and some form of motion mechanism (see e.g., [10], [36]). Hence, the investigation into the computational characteristics of PM systems has assumed again a central role, driven by the applied perspective. In fact, such systems find a plethora of natural applications in various contexts, including smart materials, ubiquitous computing, repairing at microscopic scale, and tools for minimally invasive surgery. Chiefly, large part of such investigation has been dedicated to modeling issues for effective algorithm design, performance analysis and study of the feasibility of foundational tasks that are relevant to PM. Example of robotic approaches related to PM can be found in [29], [30], [35].

The majority of previous works on models for PM have been inspired by physical systems or biological colonies [31], [34]. Among them perhaps the most promising (in terms of quality of the abstraction, as well documented in the literature [13], [20]) is the so-called geometric *Amoebot* model [19], which is inspired by the behavior of the amoeba. Such a model, as well as other prominent ones, considers a swarm of decentralized autonomous self-organizing particles that: i) are modeled as finite state automata; ii) are all identical, executing the same algorithm based on local observation of the surroundings; iii) are displaced in the cells of a hexagonal grid (represented by a triangular lattice); iv) can move from cell to cell by repeatedly alternating between two states, namely *contracted* (a particle occupies one cell) and *expanded* (the particle occupies two neighboring cells). Given these elementary features, it is rather hard to design distributed algorithms even for basic tasks (e.g. coating [13], [18], bridge building [3], shape formation [6], [19], [20], [38], shape recovery [21], and compression [6]). Very recently, programmable matter based on the Amoebot model has been exploited also in the context of recognition and reconfiguration of lattice-based cellular structures tasks related to space missions [24], [33].

Essentially all existing solutions, to solve foundational problems in this setting, have resorted to add various features to overcome the underlying hardness. Examples include:

- presence of a global serialized synchronizer: no two neighboring particles are assumed to be simultaneously activated (also known as sequential scheduling, a single particle activation at the time);
- communication: particles are endowed with the ability to explicitly interact with neighboring particles by means of shared memory, read/write operations or message passing mechanisms;
- atomicity of actions: particles are assumed to be able to perform various operations in an atomic step (e.g. observation of the surrounding, elaboration, and communication happen simultanously);
- randomization: particles can rely on random outcomes to take decisions and in particular for symmetry breaking.

These additional assumptions appear quite unrealistic, given the inherently weak and asynchronous nature of PM. Specifically, theoretical models differ in many aspects from the real systems that have been deployed and tested in the laboratories [5], [7], [28] both in terms of particles' capabilities and in how they interact. Therefore, the provided abstraction might result in being ineffective for algorithm design and performance analysis in these scenarios. On a related line of research, it is worth observing that similar modeling issues have been encountered by studies on multi-robot computing systems [2], [15]–[17], [25], [27], and metamorphic robots [32].

In this paper, we try to bridge the gap existing between theoretical studies and practice by proposing the SILBOT model, a new modeling approach for systems of programmable particles. The new abstraction inherits some features of the most established models for PM but it is closer to real-world particles systems, since it shares some peculiarities (e.g. absence of explicit communication or full asynchronicity) of the consolidated *Look-Compute-Move* model [25] for robotic swarms, extensively studied in distributed computing and widely adopted in practice. Although this paper explores theoretical issues related to the modeling and algorithm design for PM, we move a step closer to practically implementable programmable matter. In fact, the fewer capabilities are assumed the larger is the range of applicability of the results. Moreover, a weaker model allows to build safer, more energy-efficient and fault-tolerant systems. Thus, it is important to develop algorithms that require as few assumptions as possible, and to understand the limit below which tasks become unfeasible.

The main characteristics of the SILBOT model can be summarized as follows. Particles are *silent*, that is there is no explicit means of inter-particle communication. Moreover, they act in a *fully asynchronous* manner, without any restriction on simultaneous activations, on when a particle is activated or on the duration of computations or actions. As in any asynchronous distributed system we assume that each particle is activated within finite time and infinitely often. Note that without such an assumption an adversarial scheduler may prevent any algorithm from ever achieving any goal (see [25] and references therein). Finally, concerning motion, particles move similarly to the Amoebot model: a particle p is EXPANDED if it occupies one cell u and one incident edge (u, v), rather than the two adjacent cells u and v as in the Amoebot model. Then, p may switch from the EXPANDED state to the CONTRACTED one simply by occupying only cell v. The expansion toward a cell v is seen as a commitment to move

to v. The move is finalized at any subsequent activation of the particle, when the particle contracts to v, if v is empty. Clearly, more than one particle may try to expand toward the same cell v, along two different edges adjacent to v. In this case only one succeeds, and this is chosen in a classic adversarial fashion: the adversary schedules, arbitrarily, a corresponding movement. In other words, if a particle expands toward an empty cell and this, in the meantime, becomes occupied, the movement is not realized and the particle remains in the EXPANDED state. Note that the personification of an adversary is a standard tool in distributed computing to represent any possible execution of an algorithm with respect to parameters that cannot be controlled by the designed algorithms.

By the above, one could immediately observe that the SILBOT model is much weaker, computationally speaking, than other models for PM, since: i) particles have much less capabilities with respect to previous models; ii) at the same time, SILBOT captures the fully asynchronous nature of real-world PM systems. This, on the one hand, can be considered a positive feature, since it is well known from other studies on distributed computing that weaker models simplify performance analysis and lead to more reliable implementations [25]. On the other hand, as previously remarked, less computational power usually lead to harder algorithm design and even impossibility results for basic tasks [20]. However, we show that this is not the case, since the SILBOT model is powerful enough to allow the resolution of various problems that are relevant in the context of PM. In more details, we provide distributed algorithms to implement, under the SILBOT model, perhaps the most prominent primitive for PM, namely *Leader Election*. The *Leader Election* problem, besides being of theoretical interest on its own, can be considered as one of the foundational problems in systems of programmable particles [4], [19], [20], [23], [26], as its resolution is often necessary to solve more complex tasks, e.g. coating or shape formation.

Several strategies have been used in the past for leader election: in [14], [19], for instance, particles resort to randomization to break symmetries; in [4], [14], [19], [26] chirality is assumed, that is a globally consistent circular orientation of the plane shared by all particles; in [23], the scheduler is assumed so as only one particle at the time is activated. Note that, unless randomization [14], [19] or sequential scheduling [23] is assumed, the election of one single leader is not possible. Indeed, in [4] and [20] up to six and three leaders, respectively, can be elected due to possible symmetries. In what follows, therefore, we focus on this latter leader election problem, where up to three mutually neighboring particles can be elected; these form a team of leaders that may then act together to lead the other particles. We provide a deterministic algorithm for solving the problem when the initial configuration is simply connected as in [20]. Indeed we show how to emulate the erosion process presented in [20] by a simpler set of rules. Our new algorithm works in $O(n)$ rounds, with n being the number of particles in the system and a *round* being the time within which all particles have been activated at least once (this concept will be better defined later).

When the initial configuration is not simply connected and contains *holes*, that is when there is a region of the lattice delimited by particles whose interior contains empty cells, we formally show that leader election is impossible if the system needs to maintain connectivity among particles. To deal also with arbitrary connected configurations we strengthen the particles with the capability of detecting whether an empty neighboring cell is part of a hole or it belongs to the exterior lattice region. In fact, from a practical point of view, it is possible for PM particles to acquire such a knowledge by just local sensing of the surroundings (e.g. via light or pressure sensors), and without any form of global visibility or any mean for message exchange. We thus provide an algorithm for achieving *compaction of holes*, that is to convert any arbitrary connected configuration to a simply connected configuration. We also show how to achieve leader election while performing the compaction within $O(n^2)$ rounds.

Our algorithms are deterministic, and require only two states for the particles. In other words, the particles have no additional memory, except the ability to be in two (visually) distinguishable states (either EXPANDED or CONTRACTED). Finally, we formally prove that all our algorithms are correct: starting from an arbitrary initial connected configuration (either simply connected or not), we show that in the system there are at most three particles elected as leader.

II. THE SILBOT MODEL

In this section, we present the new SILBOT model for PM, where particles act independently of each other, without explicit communication, in a fully asynchronous way, based only on local knowledge. The model is built upon the well established geometric Amoebot model, one of the most popular in the literature ([4], [14], [19], [20], [23], [26]).

The Operating Environment. The particles operate on an infinite triangular lattice (representing the described hexagonal grid) embedded in the plane, where each node has 6 incident edges: nodes correspond to hexagonal cells and each edge represents a boundary shared by two cells, and each node can contain at most one particle. There are n particles in the system and at the beginning the set of n nodes containing particles induces a connected subgraph of the lattice.

Particles and Configurations. Each particle is an automaton with two states, CONTRACTED or EXPANDED (they do not have any other form of persistent memory). In the former state, the particle occupies a single node of the lattice while in the latter, the particle occupies one single node and one of the adjacent edges. Hence, a node may contain at most one particle at any time.

Each particle can sense its surroundings up to a distance of 2-hops i.e., if a particle occupies a node v, then it can see the neighbors of v and the neighbors of the neighbors of v. Specifically, a particle can determine (i.e. sense) if a node

is empty or occupied by a CONTRACTED particle, or occupied by an EXPANDED particle, for each node in its 2-hop visibility range. Hence nodes occupied by CONTRACTED particles can be distinguished from nodes occupied by EXPANDED particles. Similarly, the particle can sense within its visibility if an edge is occupied by an EXPANDED particle. In our model particles do not have any explicit means of communication. Thus, a particle can acquire information about its surroundings only by its limited vision without communications, rather using direct sensing, e.g. weak electromagnetic fields or radars.

Any positioning of CONTRACTED or EXPANDED particles that includes all n particles composing the system is referred to as a *configuration*.

Movement and States. Each particle can occupy only one node v at a time. In order to move to a neighboring node u, the particle expands on the edge between node v and node u. Thus, in EXPANDED state, the particle occupies one node and one edge (the physical interpretation is that the particle is occupying one hexagonal cell completely and has partially entered into the adjacent cell). Note that node u may still be occupied by another particle. If the other particle leaves node u in the future, the expanded particle will contract into node u during its next activation. There might be arbitrary delays between the actions of these two particles, while the connectivity is still maintained. For example, when the particle at node u has moved to another node, the edge between v and u is still occupied by the original expanded particle. In this case we say that node u is *semi-occupied*. We ensure that the set of occupied and semi-occupied nodes induces a connected configuration at all times during the execution of the proposed algorithms.

A particle commits itself into moving to node u by expanding in that direction, and at the next activation of the same particle, it is constrained to move to node u, if u is empty. A particle cannot revoke its expansion once committed.

Interaction between Particles. Our model requires no explicit communication between the particles. Inspired also by *Cellular Automata* systems (see, e.g. [1], [22]), each particle can sense the presence of other particles (CONTRACTED or EXPANDED) in its neighborhood. Ideally we would like the particles to operate with 1-hop visibility (limited to immediate neighboring nodes).

However, in order to avoid some well-known deadlock condition (see [13], [20], this will be better clarified later in the paper), particles need to acquire information about the neighbors of their neighbors, so we assume *2-hop visibility*. This means that the particles can sense nodes within 2 hops to determine the presence of CONTRACTED or EXPANDED particles.

Note that this capability is much weaker than explicit communication with neighbors (used e.g. by the Amoebot model) which allows particles to obtain information up to any arbitrary distance via multi-hop communication. Indeed some of the algorithms for Amoebots rely on explicit communication over long chains of particles [20], [23], which is not practical due to lack of robustness against failures or faults. Moreover, for small nanoscale particles, it would be infeasible to have such sophisticated communication primitives. We envision programmable particle systems where each particle acts independently based on observation of its surroundings using a set of simple rules.

Asynchrony and Rounds. The SILBOT model introduces a fine grained notion of asynchrony with possible delays between observations and movements performed by the particles..[1] All operations performed by the particles are non-atomic: that is, there can be delays between the actions of sensing the surroundings, computing the next decision, executing the decision (i.e., change of state, movement, expansion, contraction).

We make no assumptions nor restrictions on the scheduling of these events; thus any possible execution of an actual physical system can be captured by our model. This has important consequences for computability of the particle systems and requires more rigorous techniques for proving correctness of the algorithms. In particular, algorithms for this model must be inherently simple with a few rules, since this already provides an uncountably large number of possible execution sequences. We call a *round* the time within which all particles have been activated and concluded their activation time at least once. Clearly, the duration of a round is finite but unknown and may vary from time to time. Moreover, the duration of different rounds might be completely different and different rounds might also overlap in time.

We assume the well-established fairness assumption that each particle would be activated infinitely often in any infinite execution of the particle system. Due to the asynchronous nature of the system, it may happen that a particle decides (or is forced, in case of contraction) at time t to take an action, and that this action will actually be executed at time $t' > t$, when other particles might have changed their state; in other words, the action executed at time t' might be based on the obsolete observation of the surrounding taken at time t. When this happens, we refer to such an action as a *pending* action. The time required to accomplish an action is finite but unbounded. Hence a round is the shortest time period during which each particle has performed an action (where the action could be the null action if a contracted particle decides to remain contracted).

Orientation and Chirality. We do not make any additional assumptions about orientation and handedness; the particles do not agree on *chirality*, i.e., on the clockwise or counter-clockwise directions. Some papers on Amoebots assume chirality (e.g., [4], [18], [19]), while the results in [20], [23] solve the leader election problem without assuming it. In fact, in the latter paper the authors provide a procedure for the particles to agree on the clockwise direction. However, this strategy requires a lot of communication between the particles and, thus, it is not applicable in the model used here.

[1] Somewhat similar to the so-called ASYNC model for mobile robots computing (see, e.g., [8], [9]).

Overall, our algorithms work without chirality and require no other forms of orientation.

Randomness. We never assume the possibility of using randomness: particles take deterministic decisions and do not have access to random numbers. Previous results on Amoebots use randomness while some recent papers consider deterministic Amoebots [4], [20], [23]. Clearly, the deterministic model is less powerful. Moreover, practically speaking, since the individual particles are quite small, it is realistic that they may not contain any source of randomness. Particles do not have persistent memory (except for the state information) and have no explicit means of communication with each other. Each particle may be activated at any time independently from the others. Once activated, a particle looks at its surrounding (i.e., at its neighbors and at the neighbors of its neighbors) and, on the basis of such observation, decides (deterministically) its next *action* as follows.

1) A particle in CONTRACTED state occupying node v, when activated may change to EXPANDED state thus occupying node v and one of the edges leading to a neighboring node w, if the edge (v, w) is unoccupied. In this case, we say that the particle *expands* along edge (v, w);
2) A particle in EXPANDED state occupying node v and edge (v, w), when activated, always *contracts* to w (i.e., moves to node w changing its state to CONTRACTED), if w is not occupied. If w is occupied, then the particle does not change state or location. In other words, a particle in EXPANDED state is obliged to *contract* as soon as it is activated, if the destination node is empty at that time.

If two contracted particles decide to expand on the same edge simultaneously, exactly one of the particles (arbitrarily chosen by the scheduler) succeeds. If two particles are expanded on two distinct edges incident to the same node w, and both particles are activated simultaneously, exactly one of the particles (again, chosen arbitrarily by the scheduler) contracts to node w, while the other particle does not change state.

Connectivity. An important property we preserve is that the set of particles never gets disconnected. However, when particles are moving in a sequence, one following another, the leading particle must vacate a node before the particle behind it can move there. Note that in the SILBOT model there would require a synchronization mechanism between the particles. Instead we allow the particles to move asynchronously while maintaining a ''relaxed'' sense of connectivity. When a particle is in EXPANDED state occupying node v and edge (v, w), we say that node w is *semi-occupied*. Throughout this paper, we say that a configuration is *connected* if the set of nodes that are either occupied or semi-occupied form a connected subgraph of the lattice. All algorithms in this paper always maintain a connected configuration of the system of particles.

Given a triangular lattice G, a subgraph of G is *simply connected* if the envelope of its standard planar embedding has only one exterior boundary and no interior boundaries. A configuration is simply connected if the subgraph of lattice G induced by the nodes occupied by particles is simply connected. A configuration is *deeply* simply connected if it is simply connected and the nodes occupied by the *contracted* particles induce a simply connected graph.

Leader Election and Compaction Problems. We assume the system is initially in a connected configuration where all particles are CONTRACTED. We define the problem of leader election as in [20] where at most 3 particles may be elected. We remark the election of a single particle is possible only if either randomization or a sequential scheduler are employed [23]. We say a particle p recognizes itself to be a LEADER if it is CONTRACTED and, within its visibility range, there are at most two other CONTRACTED adjacent particles that are also adjacent with p. Note that to decide to be LEADER, particles need to acquire information about the neighbors of their neighbors [20]. Hence, without communication, 1-hop visibility is not enough for a particle to decide whether it is a LEADER or not. A particle recognizes to be NON-LEADER if it is EXPANDED.

Definition 1 (Particle Leader Election (PLE) Problem): An algorithm solves the Particle Leader Election (PLE) problem if the following conditions hold: i) once the algorithm terminates there are exactly one, two, or three mutually adjacent particles that are LEADER and ii) all particles are either LEADER or NON-LEADER.

Initially, all particles are CONTRACTED and represent potential candidates to become LEADER. Once the election algorithm successfully terminates, there are at most 3 mutually adjacent leaders.

In what follows, we denote by Π the set of simply connected configurations with only one, two, or three contracted particles such that the nodes occupied by the contracted particles induce a complete graph. Moreover, given a particle p, we call $N_1(p)$ and $N_2(p)$ the set of nodes that are occupied by CONTRACTED particles at distance 1 and 2, respectively, from p. Abusing notation we sometimes denote as p the node occupied by particle p. Given a set of nodes U, we call $G(U)$ the subgraph of G induced by U.

Finally, we define the following:

Definition 2 (Compaction Problem): Given an initial arbitrary connected configuration, an algorithm solves the Compaction (of holes) problem if it brings the particles into a simply connected configuration.

III. PLE WITH SIMPLE CONNECTIVITY

In this section, we present a solving algorithm for PLE when the initial configuration is simply connected, i.e., it does not contain *holes*. Alternatively, the algorithm works for any deeply simply connected initial configuration, even if not all particles are contracted. The pseudocode of the algorithm, called LESC, is reported in Algorithm 1: it is described from the point of view of a single particle.

Let us consider the standard planar embedding of the subgraph of lattice G induced by the nodes occupied by CONTRACTED particles, and the envelope (concave hull) that

Algorithm 1 Algorithm LESC (PLE With Simple Connectivity)

Require: Each particle is CONTRACTED, the set of nodes containing particles induces a simply connected subgraph of G.

Ensure: A configuration in Π.

1: **if** (p is CONTRACTED) $\wedge$ ($G(N_1(p))$ is connected) $\wedge$ ($G(N_1(p) \cup N_2(p) \cup \{p\}) \notin \Pi$) **then**
2: **if** $N_1(p) = \{q\} \vee N_1(p) = \{q, r\}$ **then**
3: Expand along (p, q);
4: **if** $N_1(p) = \{r, q, s\}$ **then**
5: Let q be the central neighbor;
6: **if** $|N_1(r)| > 2 \wedge |N_1(q)| > 3 \wedge |N_1(s)| > 2$ **then**
7: Expand along (p, q);

contains all the nodes in this embedding. The goal of the algorithm is to shrink this envelope by expanding the particles that are on its border toward the interior. Moreover, the algorithm expands the particles toward occupied nodes in such a way that no EXPANDED particle will contract again. In other words, once expanded, a particle will not compete anymore to become a leader. More precisely, the algorithm allows to expand only CONTRACTED particles p such that $G(N_1(p))$ is connected and $|N_1(p)| \leq 3$. In particular, if there exists a CONTRACTED particle p such that $G(N_1(p))$ is connected and $|N_1(p)| \leq 2$, then p expands along edge (p, q), where q is a CONTRACTED neighboring particle of p (see Line 3).

If there is a CONTRACTED particle p such that $G(N_1(p))$ is connected and $|N_1(p)| = 3$, then p expands along edge (p, q), with q being the central neighboring particle of p, only if the other two neighbors have degree at least 3 and q has degree at least 4 (see Lines 4–7). This strategy avoids that p expands while one of its neighboring particles is expanding toward p itself. We remark that 2-hop visibility is necessary to detect this special condition (as observed in [20]). Notice that no contractions are induced by Algorithm 1.

A Run of Algorithm LESC. In Figure 1 we show an example of an execution of Algorithm LESC. CONTRACTED particles are represented by black circles, whereas EXPANDED particles are represented by ellipsoids occupying one node and one edge. Circled black circles represent CONTRACTED particles currently activated.

Figure 1 (a) shows an initial configuration where six particles have been activated: according to Algorithm LESC they expand as shown in Figure 1 (b). For the ease of reading we are not providing pending actions in the example. So, from Figure 1 (c) to Figure 1 (j), we show the evolving of the configuration according to the scheduled activations of particles. We did not report any activation of EXPANDED particles as those would not lead to any change in the configuration (EXPANDED particles never change state).

In Figures 1 (c), (e) and (g) there are some activated particles that are not EXPANDED in the subsequent configurations (d), (f), and (h), respectively. This is due the priority provided by Algorithm LESC to expand first neighboring particles of smaller degree (whose direct neighborhood of CONTRACTED particles is smaller). Figure 1 (j) shows the case where only one particle remains CONTRACTED: it is the LEADER. Note that, according to Algorithms LESC, it would have been possible, with a different activation schedule, to elect up to three leaders.

Correctness of Algorithm LESC. We show that, in any deeply simply connected configuration, there always exists (at least) a particle that can expand, until the leader(s) is elected. Moreover, during the execution of the algorithm, all configurations are guaranteed to be deeply simply connected, while the number of CONTRACTED particles decreases, until a configuration in Π is reached.

To show this, we model an execution of Algorithm LESC as a path in a directed graph $H = (V, E)$, where the vertices in V correspond to any deeply simply connected configuration with at most n CONTRACTED particles, and the edges in E correspond to transitions among configurations as determined by Algorithm LESC. In particular, each vertex $u \in V$ corresponds to a configuration C_u, and there is a directed edge $(u, v) \in E$ if there exists an execution of Algorithm LESC that leads from C_u to C_v, without generating in between further configurations different from C_v.

We distinguish two types of edges: (u, v) is an *expansion edge* if, considering C_u as the initial configuration, there is a schedule in which Algorithm LESC leads from C_u to C_v in one round. Edge (u, v) is a *pending edge* if it models the transition from a configuration C_u to another configuration C_v that may occur not because one or more particles are performing expansions dictated by the algorithm from C_u, but because such particles have started their computations from some configurations different from C_u and, due to asynchrony, C_u has been generated in the meanwhile. The expansions that determine, from C_u, a pending edge are called *pending expansions*. The next theorem exploits graph H to show that Algorithm LESC converges to a configuration in Π.

Theorem 1: Starting from a deeply simply connected configuration, Algorithm LESC terminates after $O(n)$ rounds in a configuration in Π. Moreover, any configuration generated during the execution of LESC is deeply simply connected.

Proof: Initially there are n CONTRACTED particles that form a deeply simply connected configuration. Let $H = (V, E)$ be the directed graph representing the executions of Algorithm LESC as defined above. We prove the theorem by showing the three following properties:

P1. Each vertex in H, excluding those corresponding to configurations in Π, has at least one outgoing expansion edge. Moreover, vertices in H corresponding to configurations in Π are connected only by pending edges as shown in Figure 2.
P2. Each expansion dictated by Algorithm LESC corresponds to an edge in H, that is the configuration obtained by any expansion dictated by LESC is deeply simply connected.
P3. Graph H is acyclic.

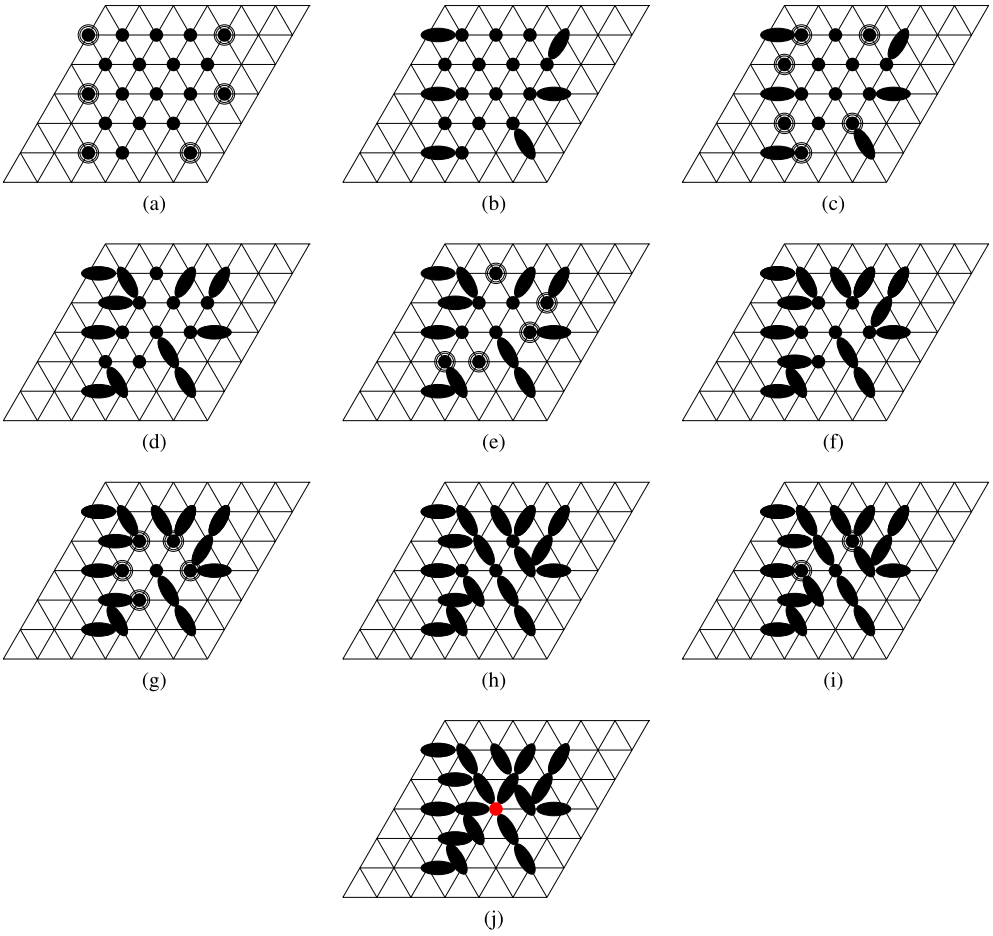

FIGURE 1. An example of execution of algorithm LESC. In this execution, at the end of the run, only one particle (the red one) is in the state LEADER.

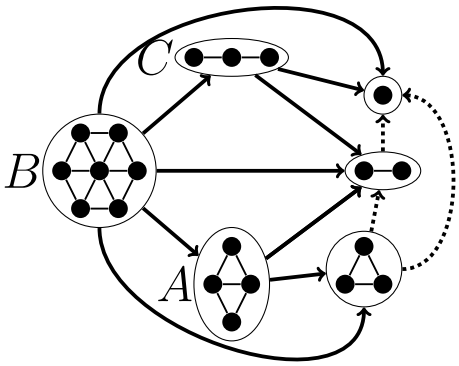

FIGURE 2. A subgraph of graph H used in the proof of Theorem 1. Each vertex is associated with a set of configurations represented by the subgraph induced by only contracted particles. Pending edges are drawn as dashed arrows, while expansion edges are drawn as bold arrows. For simplicity, the figure does not depicts all the configurations that can be obtained from configuration C according to Algorithm LESC.

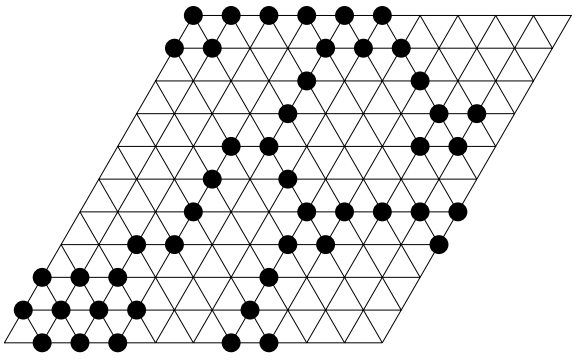

FIGURE 3. Example of a simply connected configuration.

Property P1. We first show that in any deeply simply connected configuration there exists a CONTRACTED particle p such that $G(N_1(p))$ is connected and $|N_1(p)| \leq 3$.

Let us consider the standard planar embedding of the subgraph of lattice G induced by the CONTRACTED particles and the envelope containing all the points of this embedding. Since the configuration is deeply simply connected, the shape of the envelope is a "tree of polygons", that is a set of polygons that are connected by paths of straight lines, possibly of length 0 (i.e., connected by one single point corresponding to a single particle). Moreover, the leaves of the tree are not single particles since this would imply that there is at least one particle with only one neighbor and this would be contradiction (see Figures 3 and 4 for an example).

Now, let us consider a leaf of the tree of polygons, and let us assume that it has m vertices (corresponding to m particles). Note that since this polygon is a leaf of the tree, only one of its vertices is connected to the rest of the tree and any other vertex of the polygon corresponds to a particle that has a connected neighborhood.

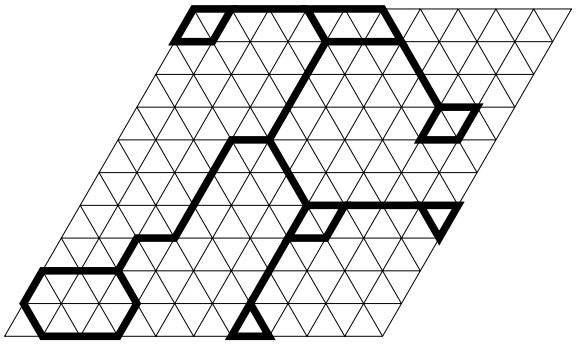

FIGURE 4. Example of tree of polygons, corresponding to the configuration in Figure 3, used in the proof of Theorem 1 for Property P1.

By contradiction, let us assume that each particle in the polygon, excluding the one that connects the polygon to the rest of the tree, has at least 4 occupied neighbors. Then the interior angle of each vertex in the polygon corresponding to these particles measures at least π. Therefore, the sum of the interior angles of the polygon is at least $(m-1)\pi$, which is a contradiction since it is known that such a sum equals $(m-2)\pi$ in any polygon. Hence, we can conclude that there must exist a CONTRACTED particle p such that $G(N_1(p))$ is connected and $|N_1(p)| \leq 3$. Furthermore, such a particle belongs to a leaf of the tree of polygons.

Next we show that, given the existence of such a particle, in any configuration not in Π there exists at least one particle that decides to expand according to Algorithm LESC.

In particular, if $|N_1(p)| \leq 2$, then p will expand (see Line 3).

If $|N_1(p)| = 3$, then p is allowed to expand only if the condition at Line 6 is satisfied. Assume that this condition is not satisfied and let r, q and s be the three neighbors of p, with q being the central one. If $|N_1(r)| \leq 2$ ($|N_1(s)| \leq 2$, resp.), then r (s, resp.) will expand, as imposed by Line 3. By referring to Figures 5 (a) and (b), if $|N_1(r)| > 2$ and $|N_1(s)| > 2$, then r shares neighbor q with p and has a further neighbor, say t_r. Similarly, s is adjacent to p, q, and has a further neighbor t_s. Then two cases can occur: if t_r and t_s are not neighbors of q, then we have a contradiction as the considered polygon is not a leaf of the tree (see Figure 5 (a)). Otherwise, if at least one among t_r and t_s is a neighbor of q, we have that $|N_1(q)| > 3$ and the condition at Line 6 is satisfied (see Figure 5 (b)), hence p will expand, which is again a contradiction.

Therefore, we can conclude that there is at least one particle that decides to expand according to Algorithm LESC. If more than two particles decide to expand along the same edge, at least one of them will succeed according to the model.

Finally, it remains to show that the vertices corresponding to configurations in Π are connected as shown in Figure 2. Observe that if there are no pending expansions, and the configuration is in Π, then no particle is expanded by Algorithm LESC. It follows that edges among vertices corresponding to configurations in Π can only be pending edges. Hence, Property P1 follows.

Property P2. By contradiction, let us assume that after an expansion (possibly pending) dictated by Algorithm LESC, the configuration obtained is not deeply simply connected anymore. In the following, we denote by $N_1^t(p)$ the set $N_1(p)$, as seen by a particle p if it is activated at some time t. Let t be the first time instant when the subgraph G_t of lattice G, induced by the CONTRACTED particles, is not simply connected. We distinguish the two possible cases:

1) G_t **has a hole.** We first show that an EXPANDED particle does not contract again. By contradiction, let us consider the first EXPANDED particle p that contracts. Let (v, u) be the edge where p is EXPANDED, with u being the node on which p contracts after. According to the algorithm, node u was occupied when p decided to expand, while it has to be empty when p decides to contract. Thus, the particle that was occupying node u has been CONTRACTED between the decision of expansion and the contraction of p, which is a contradiction as p is the first particle that contracts after an expansion.
Therefore, to create a hole the only possibility is that a CONTRACTED particle p has decided to expand at some time $t' \leq t$ and it is actually EXPANDED at time t, surrounded by 6 CONTRACTED particles. Since EXPANDED particle do not contract again, function $|N_1^t(p)|$ is non-increasing with respect to the time t. This implies that $|N_1^{t'}(p)| \geq |N_1^t(p)| = 6$, and p decided to expand at time t', with $|N_1^{t'}(p)| = 6$. This is in contradiction with Algorithm LESC that allows a particle p to expand only if $|N_1(p)| \leq 3$. Hence, Property P2 follows.
2) G_t **is disconnected.** If a particle p becomes EXPANDED at time t and G_t becomes disconnected because of such an expansion, then either $G(N_1(p))$ was disconnected when p decided to expand or a neighboring particle of p expanded at a time between the time when p decided to expand and t.
The former case contradicts Algorithm LESC that allows p to expand only if $G(N_1(p))$ is connected. For the latter

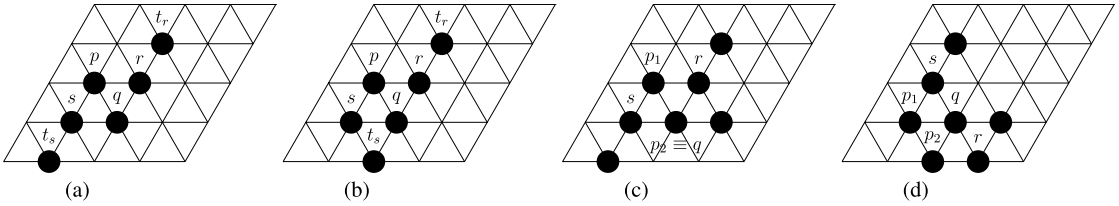

FIGURE 5. Configurations used in the proof of Theorem 1.

case, let us assume that the disconnection at time t is due to two particles p_1 and p_2 that are CONTRACTED, occupy two neighboring nodes at some time before t, and that are both EXPANDED at time t (they may expand at different times, in any order, and t is the time when the second particle is EXPANDED). Let us denote as t_1 and t_2 the time when p_1 and p_2 decided to expand, respectively, and let us assume without loss of generality that $t_1 \leq t_2 \leq t$.

If t_1' and t_2' denote the time when p_1 and p_2 actually expand, respectively, then we must have that $t_1' \geq t_2$ as otherwise the expansions of p_1 and p_2 are sequential (i.e., we are in the former case). Therefore we have two possible cases: either $t_1 \leq t_2 \leq t_1' \leq t_2' = t$ or $t_1 \leq t_2 \leq t_2' \leq t_1' = t$. We also assume that at times t_1 and t_2 the configuration is not in Π, as otherwise p_1 and/or p_2 will not decide to expand according to Algorithm LESC. Since $|N_1^{t_1}(p_2)| \geq |N_1^{t_2}(p_2)|$ and, according to Algorithm LESC, a particle p is allowed to expand only if $G(N_1(p))$ is connected and $|N_1(p)| \leq 3$, we now analyze the four possible cases:

a) $|N_1^{t_1}(p_1)| = 1$ In this case p_1 expands toward p_2, and p_2 can decide to expand only after p_1 is EXPANDED otherwise $G(N_1(p_2))$ is not connected. Therefore, we have $t_1' < t_2$ which is a contradiction.

b) $|N_1^{t_1}(p_1)| = 2$ Since, before t, particles p_1 and p_2 occupy neighboring nodes and $G(N_1^{t_1}(p_1))$ is connected, then there must be a third particle p_3 located at a node that is adjacent to both p_1 and p_2. According to Algorithm LESC, p_1 must be EXPANDED along (p_1, p_2) or along (p_1, p_3). Now, if p_3 has two neighbors only (namely p_1 and p_2), then p_2 can decide to expand only after that both p_1 and p_3 are EXPANDED as otherwise $G(N_1(p_2))$ is not connected. This is a contradiction as it implies that $t_1' < t_2$. It follows that the only possible case is p_3 having other neighbors, different from p_1 and p_2, and hence any expansion of p_2 toward any of its occupied neighbors cannot disconnect G_t.

c) $|N_1^{t_1}(p_1)| = 3$ and $|N_1^{t_1}(p_2)| \leq 2$ In this case, p_1 cannot expand at time t_1 according to Algorithm LESC.

d) $|N_1^{t_1}(p_1)| = 3$ and $|N_1^{t_1}(p_2)| \geq 3$ We have two cases: p_2 is the central node in $N_1^{t_1}(p_1)$ (i.e., node q in Line 6) and there are 2 occupied nodes, s and t, that are neighbors of both p_1 and p_2 (see Figure 5 (c)), or p_1 and p_2 share a common central neighbor q and have two further different occupied neighbors, s and r (see Figure 5 (d)).

In the first case, p_1 decides to expand along edge (p_1, p_2). Since $|N_1^{t_1}(p_1)| = 3$ and p_1 will not expand before t_2, then according to Algorithm LESC, p_2 is not allowed to expand until some other particles expand. Moreover, again since $t_1' \geq t_2$, the only chances for p_2 to decide to expand at time t_2 are that $\{p_1, r\} \subseteq N_1^{t_2}(p_2) =$ (i.e., particle s is EXPANDED), in which case p_2 expands toward p_1 or r; or, symmetrically, that $\{p_1, s\} \subseteq N_1^{t_2}(p_2)$ (i.e., particle r is EXPANDED), in which case p_2 expands toward p_1 or s. In either case the expansion of p_1 and p_2 does not disconnect G_t.

In the second case, according to Line 7, p_1 will expand along edge (p_1, q), while p_2 will expand along one of its neighbors. In any case G_t is not disconnected as node q in the time interval between t_1 and t can only expand toward one of the nodes in $(\{p_1, p_2\} \cup N_1^{t_1}(p_1) \cup N_1^{t_1}(p_2)) \cap N_1^{t_1}(q)$. In particular, observe that in any time of this interval, at most one among p_1 and p_2 can be EXPANDED, and depending on the way in which the other nodes in $N_1^{t_1}(q)$ expand, q can expand because it has one neighbor (either p_1 or p_2), two neighbors ($\{p_1, p_2\}$, $\{p_1, s\}$, or $\{p_1, r\}$), or three neighbors ($\{p_1, p_2, s\}$ or $\{p_1, p_2, r\}$). In any of these cases the resulting graph G_t is not disconnected. In any other case $G(N_1(q))$ would be disconnected and so q will not expand.

Hence, Property P2 follows.

Property P3. First observe that, for each directed edge $(u, v) \in E$, the number of CONTRACTED particles in C_v is smaller than the number of CONTRACTED particles in C_u. In fact, when an expansion is performed, at least one CONTRACTED particle in C_u becomes EXPANDED in C_v toward an occupied node belonging to the interior of the configuration.

Moreover, by Property P2, EXPANDED particles cannot contract (they remain EXPANDED toward occupied nodes during the whole execution). Therefore, we can define a topological ordering on the vertices of H as a linear extension of the partial ordering given by the number of CONTRACTED particles of the corresponding configurations. It follows that H is acyclic, hence P3 follows.

We are now ready to prove the statement of the theorem. First, recall that the only sink vertices of H w.r.t. expansion edges (i.e., the only vertices that have no outgoing expansion edges) are those corresponding to configurations in Π. Therefore, starting from any configuration C_u, an execution of Algorithm LESC corresponds to a directed path in H that starts at u and finishes, after a finite number of edges (i.e., expansions), in a vertex corresponding to a configuration in Π. Clearly, the final vertex of the path depends on the exact schedule, which determines the expansions that are pending and, hence, the pending edges in the path.

Finally, by Property P3, any path in H that starts from the vertex corresponding to the initial configuration and ends in a vertex corresponding to a configuration in Π, has a length at most equal to the number of initially CONTRACTED particles. Thus, the algorithm converges in at most n rounds, and the theorem follows. ■

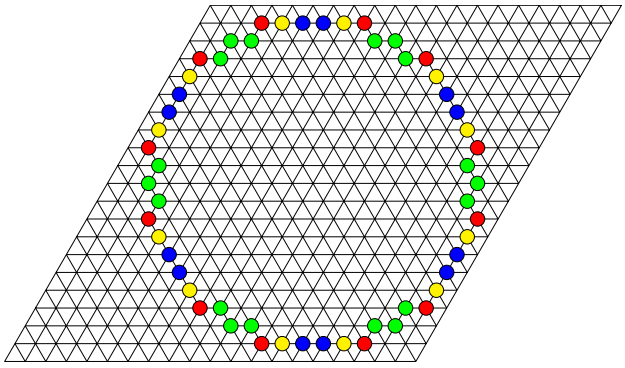

FIGURE 6. A connected configuration with a hole, used in the proof of Theorem 2. Colors identify particles having the same 2-hop neighborhood.

IV. ARBITRARY CONNECTED CONFIGURATIONS

In this section, we show that, if PLE has to be solved while preserving connectivity, then the initial configuration must be simply connected, otherwise further assumptions become necessary.

Theorem 2: Starting from an arbitrarily connected configuration, PLE cannot be solved without disconnecting the set of particles. This holds even if the particles are endowed with unlimited memory and chirality.

Proof: Consider the not simply connected configuration of particles shown in Figure 6. For the sake of our analysis, we assigned the same color to particles having the same 2-hop neighborhood (colors have no other purpose and all particles are identical). Since any decision taken by any particle depends on its local neighborhood, particles having the same color also behave the same, if activated concurrently.

Clearly, starting from the configuration in Figure 6, PLE cannot be solved if all particles maintain their positions. This is evident due to the initial symmetry of the configuration which is conserved as long as no particle moves. Thus, any algorithm for PLE must instruct some particles to move (by expanding toward empty nodes).

We now claim that any algorithm that instructs a particle to move, based only on local information, may cause the configuration to become disconnected. In the following we will assume the specific execution (schedule) where any particle is activated twice in sequence, thereby eliminating any delays between the expansion and contraction of the particle. In this particular execution, moves of the particles can be considered to be atomic. We analyze moves of equivalent particles (i.e., having the same color) separately. We can do this without loss of generality, since the adversarial scheduler can force only one group to move, while all the other particles are not activated.

Let us suppose that the algorithm makes the blue particles move: any move of a blue particle to any of the four neighboring empty nodes cause the set of particles to get disconnected (it is sufficient that the scheduler forces only one particle in each group of consecutive blue particles to move). For yellow particles, a similar argument holds: no matter which of the four neighboring empty nodes is chosen as a destination, if the scheduler moves all the yellow particles at the same time, then the configuration is disconnected.

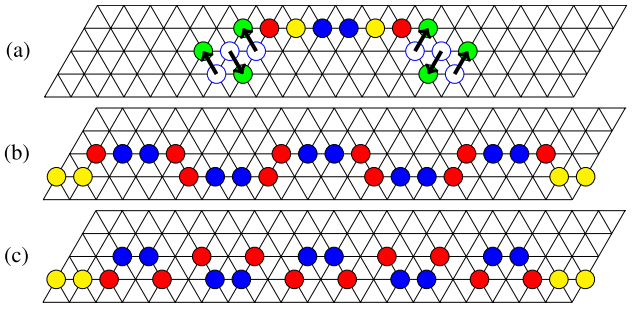

FIGURE 7. (a) The configuration after one move by green particles. Arrows depict the performed moves. (b) A connected configuration without holes. (c) The configuration obtained after one move by red particles.

For green particles, the situation is slightly different. They all have the same view irrespective of chirality, and the only possible move is toward the neighboring node that is surrounded by two other particles. Any other move clearly causes disconnection if all green particles move concurrently. If such a move is chosen, the scheduler can activate all the green particles at the same time (twice in a row). After such moves, the situation would be as shown in Figure 7 (a), where a portion of Figure 6 is reported. Hence, the resulting configuration is disconnected. Thus, the last possible choice for the algorithm is to instruct all red particles to move first to the neighboring nodes that are inside the hole (see Figure 6); in fact, this is the only move maintaining connectivity even if all the red particles are activated simultaneously.

Let us now consider a different initial configuration as shown in Figure 7 (b). Note that the red particles in this configuration have the same view as the red particles in the configuration from Figure 6. Thus, the same algorithm would instruct the red particles to move in the same way as in the previous case. Consider any two adjacent red particles: if they have the same chirality and they are activated simultaneously, then they would move in opposite directions. The resulting configuration, shown in Figure 7 (c), is one where the set of particles gets disconnected. Hence it is impossible to solve PLE without disconnecting the set of particles, and the theorem follows. ■

Note that it is not straightforward to re-establish a connected configuration once the particles disconnect. In fact, due the asynchrony of the system, there might be arbitrary delays before the particles that moved are activated again; during this time, other particles might move (they might not be aware of the disconnection). The local vision, the lack of communication between the particles and the lack of memory make it difficult to reconnect the particles, and thus achieve leader election. Furthermore, if the configuration remains disconnected, it might also be that too many leaders are elected.

V. COMPACTION AND ELECTION

Theorem 2 does not prove the impossibility of solving PLE once the configuration disconnects. On the other hand,

we also observed that it is of practical interest to maintain connectivity. To this end, we strengthen particles with a very simple capability, called *exterior awareness*, that informally is the power of detecting what is "outside" the configuration and what is "inside" (that is, holes). Formally, this capability is defined as follows:

Definition 3 (Exterior Awareness): A particle can distinguish whether a node v within its visibility range is either CONT, EXP, IN *or* OUT *where: a* CONT *node is a node occupied by a* CONTRACTED *particle; an* EXP *node is a node occupied by an* EXPANDED *particle, an* IN *node is an empty node that is part of a hole; an* OUT *node is an exterior empty node (a node that is not part of any hole).*

In what follows we will show that the particles, by just having this simple capability, are able, starting from any connected configuration (possibly having holes) to achieve *compaction of holes*, while solving PLE at the same time. Notice that, during the process, the set of particles always forms a connected configuration, thus overcoming the impossibility result of Theorem 2.

More formally, we will present an algorithm that, starting from any arbitrarily connected configuration (hence not necessarily simply), it is possible to obtain a deeply simply connected configuration (see Section II); at the same time, it also leads to elect a set of at most three leaders, as done with Algorithm LESC. This algorithm, called CHLE and whose pseudocode is reported in Algorithm 2, is again described from the point of view of a single particle. Before introducing it, we need the following further notation: given a particle p, we denote by $N_1(p, C)$ and $N_1(p, I)$ the set of CONT and IN nodes adjacent to p, resp., and by $N_1(p, IC)$ the set $\{N_1(p, C) \cup N_1(p, I)\}$.

The rationale of Algorithm CHLE is similar to that of Algorithm LESC: the goal is to shrink the envelope of the standard planar embedding of the subgraph of lattice G induced by the nodes occupied by contracted particles until we obtain a configuration in Π; the difference here is that we have to take also into account IN nodes in the envelope. In more detail, we consider the standard planar embedding of the subgraph of lattice G induced by CONT and IN nodes, and we take the envelope that contains all the nodes in this embedding (that is, the nodes occupied by contracted particles plus the holes).

To this aim, following the approach of Algorithm LESC, Algorithm CHLE allows to expand only particles p such that $G(N_1(p, IC))$ is connected and $|N_1(p, IC)| \leq 3$. However, when a node in $N_1(p, IC)$ is an empty internal node (i.e.$|N_1(p, I)| = 1$) and p is allowed to expand, then p always expands toward the internal node.

In particular, if there exists a contracted particle p such that $|N_1(p, IC)| = |N_1(p, I)| = 1$, then p expands toward the only internal node, while if $|N_1(p, IC)| = |N_1(p, C)| = 1$ it expands toward the only contracted particle (see Line 3).

If $|N_1(p, IC)| = 2$ and $G(N_1(p, IC))$ is connected, instead, then: i) if there is a node q in $N_1(p, I)$, p expands along edge (p, q) (Line 6); ii) otherwise, it expands toward any contracted particle (Line 8).

Algorithm 2 Algorithm CHLE (Compaction of Holes and PLE)

Require: A connected configuration where each particle is CONTRACTED.

Ensure: A configuration in Π.

1: **if** (p is CONTRACTED) $\wedge$ ($G(N_1(p, IC))$ is connected) $\wedge$ ($G(N_1(p, C) \cup N_2(p) \cup \{p\}) \notin \Pi$) **then**
2: **if** $N_1(p, IC) = \{q\}$ **then**
3: Expand along (p, q);
4: **if** $N_1(p, IC) = \{q, r\}$ **then**
5: **if** $q \in N_1(p, I)$ **then**
6: Expand along (p, q);
7: **else**
8: Expand along (p, r);
9: **if** $N_1(p, IC) = \{r, q, s\}$ **then**
10: **if** $\{r, q, s\} \cap N_1(p, I) = \{x\}$ **then**
11: Expand along (p, x);
12: **else if** $|N_1(r, C)| > 2 \wedge |N_1(q, C)| > 3 \wedge |N_1(s, C)| > 2$ **then**
13: Expand along (p, q);

Finally, if $|N_1(p, IC)| = 3$ and $G(N_1(p, IC))$ is connected, then we have two cases: either there is one empty internal node or all the three neighbors in $N_1(p, IC)$ are occupied by contracted particles. In the former case, p expands toward the internal node (Line 11), otherwise it expands along the central contracted neighbor only if the condition at Line 12 is satisfied. Note that a particle p with connected $G(N_1(p, IC))$, $|N_1(p, IC)| = 3$, and $|N_1(p, I)| \geq 2$ cannot exist.

Notice that, differently from Algorithm LESC which forces particles to expand toward occupied nodes only, here it may happen that a particle is expanded toward an internal empty node. For instance, let us consider a particle p that was contracted on node u and then expands along an edge (u, v), where v is an internal empty node. If the scheduler activates p when it is expanded, then p is forced to contract in v. If more then one particle tries to contract on the same internal node, only one of them (decided by the scheduler) will succeed and the other ones will remain expanded. In two steps, particle p moved from u to v and, by repeating these movements, the hole containing v is eventually filled with contracted particles.

Finally, we observe that in simply connected configurations Algorithm CHLE is equivalent to Algorithm LESC. The next theorem uses arguments similar to those used in Theorem 1 to show that Algorithm CHLE converges to a configuration in Π in a finite number of rounds. The main difference is that now the vertices of graph H represent all (not necessarily simply) connected configurations (including semi-occupied nodes) with at most n contracted particles.

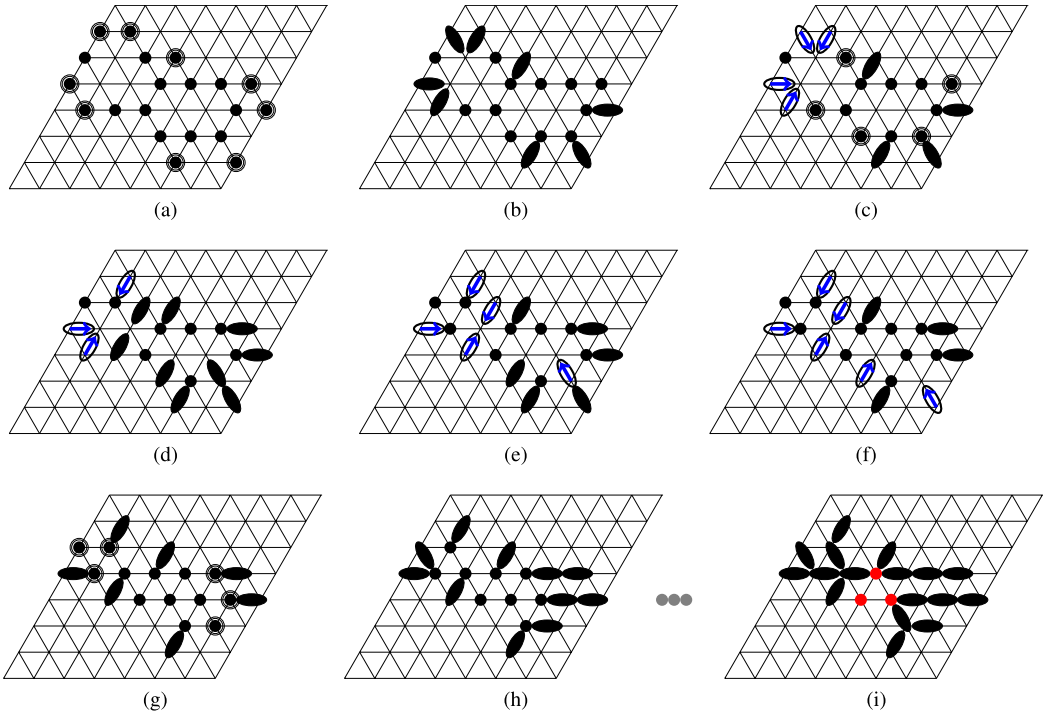

FIGURE 8. An example of execution of algorithm CHLE. Three particles in state LEADER are elected according to the schedule at hand.

A Run of Algorithm CHLE. In Figure 8 we report a possible execution of Algorithm CHLE. Differently from the example related to Algorithm LESC, here we highlight the behavior of EXPANDED particles. This is done by means of blue arrows. As shown in Figure 8 (c), (d) and (e), when two EXPANDED particles compete to move toward a same empty node, then the scheduler decides who wins. We do not show all possible configurations that from (h) lead to the final configuration (i) where three mutually neighboring leaders are elected.

Correctness of Algorithm CHLE. In what follows, we show the correctness of Algorithm CHLE, i.e. that it terminates in finite number of rounds in one of configurations of Π.

Theorem 3: Starting from any connected configuration of contracted particles, Algorithm CHLE terminates after $O(n^2)$ rounds in a configuration in Π. Moreover, any configuration (including semi-occupied nodes) generated during the execution of CHLE is connected.

Proof: We assume that, in the initial configuration, there are n CONTRACTED particles that form a (non simply) connected configuration, that is there is at least one IN node. Otherwise, the claim follows by Theorem 1: indeed, Algorithm CHLE does not need exterior awareness and behaves as Algorithm LESC whenever the initial configuration is simply connected.

We will use an argument similar to that used in the proof of Theorem 1: we model an execution of Algorithm CHLE as paths in a directed graph $H = (V, E)$. However, in this case, the vertices of the graph represent the *connected* configurations with n CONTRACTED or EXPANDED particles, and edges correspond to actions induced by Algorithm CHLE. Therefore, each node $u \in V$ corresponds to a configuration C_u, and and there is an edge $(u, v) \in E$ if there exist a schedule and an action of Algorithm CHLE that lead from C_u to C_v, without any intermediate configurations.

First, let us show the following three properties:

P1. Each node in H, but those corresponding to configurations in Π, has at least one outgoing expansion edge. Moreover, nodes in H corresponding to configurations in Π are connected only by pending edges as shown in Figure 2.

P2. Each action of Algorithm CHLE corresponds to an edge in H, i.e., the configuration obtained by any action of CHLE is connected.

P3. Graph H is acyclic.

Property P1. The difference with Property P1 proven in Theorem 1 is that here we consider the tree of polygons as induced by both IN and CONT nodes. In particular, from the proof of Theorem 1, we replace the neighborhood $N_1(p)$ of nodes occupied by CONTRACTED particles with the neighborhood $N_1(p, IC)$ made of IN or CONT nodes.

Also, let us consider the standard planar embedding of the sub-graph of grid G induced by IN or CONT nodes, and the envelope containing all nodes of such embedding in which the external nodes are connected by straight lines: the shape of the embedding is again a tree of polygons and the leaves of the tree are not straight lines as this would imply that there is at least a particle having only one IN or CONT neighbor.

It follows that there exists a CONTRACTED particle p such that $G(N_1(p, IC))$ is connected and $|N_1(p, IC)| \leq 3$ and

that such a particle belongs to a leaf polygon of the tree. Given this condition, we can use the same arguments of Theorem 1 to show that one particle expands (note that here the condition for a particle to expand is relaxed w.r.t. Algorithm LESC as a CONTRACTED particle with three connected IN or CONT neighbors, one of which is a IN node, always expands). Finally, the last part of Property P1 again follows by the same argument used in the proof of Theorem 1.

Property P2. We first observe that initially the sub-graph G_t of grid G induced by IN and CONT nodes is *simply* connected (otherwise, by contradiction, sub-graph G_t of grid G induced by CONT nodes would not be connected). Therefore, showing that each action of Algorithm CHLE corresponds to an edge in H, i.e., that the configuration obtained by any action of CHLE is connected, is equivalent to showing that the sub-graph of grid G induced by IN and CONT nodes is simply connected.

The proof follows similar lines of the proof of Property P2 in Theorem 1: the difference is that in Algorithm CHLE a particle with two CONT and one IN neighbors is allowed to expand towards the single IN neighbor without restrictions. We will now show that this degree of freedom does not disconnect the graph.

Let us assume, by contradiction, that after an action of Algorithm CHLE (possibly a pending action), the sub-graph G_t of grid G induced by such nodes is not simply connected. Let us consider the first time instant t when this occurs. We have two possible cases:

1) G_t **has a hole.** Therefore, at time t, there exists either an OUT node or an EXP node surrounded by 6 IN or CONT nodes. Thus, either a CONTRACTED particle expanded toward a OUT node, which is in contrast with Algorithm CHLE that allows a particle p to expand only toward IN or CONT nodes, hence having a contradiction; or a CONTRACTED particle surrounded by 6 CONTRACTED particles decided to expand, which is again in contrast with Algorithm CHLE that allows a particle p to expand only if $|N_1(p, IC)| \leq 3$, hence having again a contradiction. In both cases, Property P2 follows.

2) G_t **is disconnected.** If only one particle p is EXPANDED at time t and G_t becomes disconnected because of such expansion, then $G(N_1(p, IC))$ was already disconnected before t. This would be a contradiction, since Algorithm CHLE allows p to expand only if $G(N_1(p, IC))$ is connected.
Therefore, there must exist at least two nodes p_1 and p_2 such that: i) they are either IN or CONT nodes; ii) they are neighboring nodes before t; iii) they are EXP nodes at time t.
According to Algorithm CHLE, a particle p is allowed to expand only if $G(N_1(p, IC))$ is connected and $|N_1(p, IC)| \leq 3$. We distinguish five possible cases:
   a) $|N_1(p_1, IC)| = |N_1(p_2, IC)| = 3$. In this case, if p_2 is the central node in $N_1(p_1, IC)$ and it is a CONT node, then p_1 either is IN node or is a CONT node and the corresponding particle does not expand (condition at line 12 is not satisfied, since $|N_1(q, IC)| = 3$).
   Otherwise, p_1 and p_2 are both CONT nodes, the corresponding particles share a common central neighbor q, and they will expand along edges (p_1, q) and (p_2, q) without disconnecting G_t.
   b) $|N_1(p_1, IC)| = 3$ and $|N_1(p_2, IC)| = 2$. In this case, only the particle at p_2, if any, is allowed to expand according to Algorithm CHLE.
   c) $|N_1(p_1, IC)| = |N_1(p_2, IC)| = 2$. Since, before t, nodes p_1 and p_2 are neighboring and both $G(N_1(p_1, IC))$ and $G(N_1(p_2, IC))$ are connected, then there is a third node p_3, adjacent to both p_1 and p_2, with p_3 a CONT node.
   According to Algorithm CHLE, at time t, p_1 is an EXP node whose particle is EXPANDED along either (p_1, p_2) or (p_1, p_3). Similarly, p_2 is an EXP node whose particle is EXPANDED along either (p_2, p_1) or (p_2, p_3). In any case, G_t is not disconnected.
   d) $|N_1(p_1, IC)| = 2$ and $|N_1(p_2, IC)| = 1$. In this case $G(N_1(p_1, IC))$ is not connected, since the other occupied node is not adjacent to the one occupied by p_2. Therefore p_1 does not expand according to CHLE.
   e) $|N_1(p_1, IC)| = |N_1(p_2, IC)| = 1$. In this case the configuration before t is made of only two CONT neighbors, hence it belongs to Π.

Hence, Property P2 follows.

Property P3. Let us associate to each vertex u of H a pair (α_u, β_u), where α_u is the sum of the number of CONT and the number of IN nodes in C_u, while β_u is the number of IN nodes in C_u. We observe that, for each edge (u, v) of H, (α_v, β_v) is lexicographically smaller than (α_u, β_u): in fact, after an action is performed, either at least a CONTRACTED particle in C_u is EXPANDED in C_v (hence the number of CONT nodes decreases) or at least an EXPANDED particle is contracted back (hence the number of IN nodes decreases by at least $k \geq 1$ but the number of CONT increases of the same amount k). Therefore, it is possible to define a topological ordering of the nodes of H as a linear extension of the partial ordering given by the pair (α_i, β_i) of the corresponding configurations C_i. Hence, H is acyclic.

Moreover, each path in H that starts from the node corresponding to the initial configuration and ends in a node corresponding to a configuration in Π has a length bounded by the sum of the initial number of CONTRACTED particles and the initial number of IN nodes. Since the former is n and the latter is bounded by n^2, the theorem follows. ■

VI. CONCLUSION AND EXTENSIONS

In this paper we considered a weaker variant of the well-known Amoebot model, where particles have limited visibility, do not have any means of direct communication (i.e., sending of messages), can rely on just one bit of private

and persistent memory, and each particle can sense its surroundings up to a distance of 2-hops. Moreover, the system is totally asynchronous, and the actions of the particles can be executed at each particle's pace. Finally, particles can only decide on expansions, and contractions are automatic and depend on the activation schedule as decided by the adversary.

Despite the weakness of this scenario, we showed that it is indeed possible to solve the Leader Election problem: we presented two algorithms that solve the problem by deterministically electing either one, two or three leaders, in both simply connected (i.e., with no holes) and arbitrary but connected initial configurations. More specifically, for arbitrary but connected initial configurations, we presented a solution that uses Exterior Awareness capability to solve PLE without disconnecting the configuration.

First, we note that the 2-hops sensing assumption, despite being much weaker than any direct communication capability, cannnot be further relaxed to the ideal 1-hop sensing (sense only direct neighbors): in fact, to decide to be LEADER, particles need to acquire information about the neighbors of their neighbors [20].

Moreover, given the extreme simplicity of the kind of actions that can be taken by particles (either expand or contract), it is clearly necessary to endow the particles with some extra-power, in order to allow them to perform some other non-trivial task. First of all, in our model particles cannot ''undo'' an expansion move; i.e., once a particle is EXPANDED along edge (u, v), it cannot become CONTRACTED back on node u. Thus, an obvious strengthening of the model is that to add an explicit contraction move decided by the particle (and not by the scheduler, as assumed in this paper).

Also, given the asynchrony of the system, the limited visibility, and the presence of just one bit of memory, in order to be able to perform a new task after leader election it is necessary to communicate somehow the fact that a leader has been elected to all particles in the systems; i.e., adding an explicit termination phase. This can be achieved by strengthening the particles with other visible states (beside EXPANDED and CONTRACTED). In particular, once a particle recognizes itself as a leader, it might switch to a LeaderElected state, and wait until all its neighbors at distance one become either LeaderElected or EXPANDED (that is, *non-leader*). Note that these visible states can be modeled as visible lights, as done in other models investigated in literature, such as [12].

Once the leaders realize that all their neighbors successfully completed the leader election task, a new task can be thus initiated. For instance, if the initial configuration had holes, the leaders might ask all particles to contract again (starting from the ones closest to the leaders, to the furthest ones): hence, starting from a not simple (but connected) initial configuration, we can reach a simple configuration with all particles contracted. Then, from here, the leader(s) status might be ''transferred'' to the external border of the (simple) configuration, and from there the particles might start moving in lines (either one, two or three) following the just appointed leaders in order to form some pattern.

In conclusion, despite the extreme simplicity of the basic model introduced here, it is possible to perform a not trivial task such as the leader election; also, by adding a few of extra capabilities (such as a constant number of visible lights), the particles might be able to perform other interesting and non-trivial tasks, that we aim at studying in the future.

ACKNOWLEDGMENT

A preliminary extended abstract appeared in the Proceedings of the 2020 International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS).

REFERENCES

[1] A. Adamatzky, *Game of Life Cellular Automata*, A. Adamatzky, Ed. London, U.K.: Springer-Verlag, 2010, doi: 10.1007/978-1-84996-217-9.
[2] M. Amir and A. M. Bruckstein, ''Minimizing travel in the uniform dispersal problem for robotic sensors,'' in *Proc. 18th Int. Conf. Auto. Agents MultiAgent Syst.*, Richland, SC, USA, 2019, pp. 113–121, 2019.
[3] M. Andrés Arroyo, S. Cannon, J. J. Daymude, D. Randall, and A. W. Richa, ''A stochastic approach to shortcut bridging in programmable matter,'' *Natural Comput.*, vol. 17, no. 4, pp. 723–741, Dec. 2018.
[4] A. Rida Bazzi and L. Joseph Briones, ''Stationary and deterministic leader election in self-organizing particle systems,'' in *Proc. 21st Int. Symp. Stabilization, Saf., Secur. Distrib. Syst. (SSS)*, vol. 11914. Cham, Switzerland: Springer, 2019, pp. 22–37, doi: 10.1007/978-3-030-34992-9_3.
[5] A. Becker, Y. Ou, P. Kim, M. Jun Kim, and A. Julius, ''Feedback control of many magnetized: Tetrahymena pyriformis cells by exploiting phase inhomogeneity,'' in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Nov. 2013, pp. 3317–3323.
[6] S. Cannon, J. J. Daymude, D. Randall, and A. W. Richa, ''A Markov chain algorithm for compression in self-organizing particle systems,'' in *Proc. ACM Symp. Princ. Distrib. Comput.*, 2016, pp. 279–288.
[7] S. Cannon, J. Joshua Daymude, W. Savoie, R. Warkentin, S. Li, I. D. Goldman, D. Randall, and W. Andréa Richa, ''Phototactic supersmarticles,'' *CoRR*, vol. abs/1711.01327, pp. 1–9, May 2017.
[8] S. Cicerone, G. Di Stefano, and A. Navarra, ''Asynchronous arbitrary pattern formation: The effects of a rigorous approach,'' *Distrib. Comput.*, vol. 32, no. 2, pp. 91–132, Apr. 2019.
[9] M. Cieliebak, P. Flocchini, G. Prencipe, and N. Santoro, ''Distributed computing by mobile robots: Gathering,'' *SIAM J. Comput.*, vol. 41, no. 4, pp. 829–879, Jan. 2012.
[10] S. Contera, *Nano Comes to Life: How Nanotechnology Is Transforming Medicine and the Future of Biology*. Princeton, NJ, USA: Princeton Univ. Press, 2019.
[11] G. D'Angelo, M. D'Emidio, S. Das, A. Navarra, and G. Prencipe, ''Leader election and compaction for asynchronous silent programmable matter,'' in *Proc. 19th Int. Conf. Auto. Agents Multiagent Syst.*, Auckland, New Zealand, May 2020, pp. 276–284.
[12] S. Das, P. Flocchini, G. Prencipe, N. Santoro, and M. Yamashita, ''Autonomous mobile robots with lights,'' *Theor. Comput. Sci.*, vol. 609, pp. 171–184, Jan. 2016.
[13] J. J. Daymude, Z. Derakhshandeh, R. Gmyr, A. Porter, A. W. Richa, C. Scheideler, and T. Strothmann, ''On the runtime of universal coating for programmable matter,'' *Natural Comput.*, vol. 17, no. 1, pp. 81–96, Mar. 2018.
[14] J. Joshua Daymude, R. Gmyr, W. Andréa Richa, C. Scheideler, and T. Strothmann, ''Improved leader election for self-organizing programmable matter,'' in *Proc. 13th Int. Symp. Algorithms Experiments for Wireless Sensor Netw.*, vol. 10718. Cham, Switzerland: Springer, 2017, pp. 127–140.
[15] M. D'Emidio, G. Di Stefano, D. Frigioni, and A. Navarra, ''Characterizing the computational power of mobile robots on graphs and implications for the Euclidean plane,'' *Inf. Comput.*, vol. 263, pp. 57–74, Dec. 2018.
[16] M. D'Emidio, D. Frigioni, and A. Navarra, ''Characterizing the computational power of anonymous mobile robots,'' in *Proc. IEEE 36th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jun. 2016, pp. 293–302.

[17] M. D'Emidio, D. Frigioni, and A. Navarra, "Synchronous robots vs asynchronous lights-enhanced robots on graphs," *Electr. Notes Theor. Comput. Sci.*, vol. 322, pp. 169–180, Dec. 2016.

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

[19] Z. Derakhshandeh, R. Gmyr, T. Strothmann, A. Rida Bazzi, W. Andréa Richa, and C. Scheideler, "Leader election and shape formation with self-organizing programmable matter," in *Proc. 21st Int. Conf. Comput. Mol. Program.*, vol. 9211. Cham, Switzerland: Springer, 2015, pp. 117–132.

[20] G. A. Di Luna, P. Flocchini, N. Santoro, G. Viglietta, and Y. Yamauchi, "Shape formation by programmable particles," *Distrib. Comput.*, vol. 33, no. 1, pp. 69–101, Feb. 2020.

[21] G. A. . Luna, P. Flocchini, G. Prencipe, N. Santoro, and G. Viglietta, "Line recovery by programmable particles," in *Proc. 19th Int. Conf. Distrib. Comput. Netw.*, 2018, pp. 1–10.

[22] G. . Stefano and A. Navarra, "The game of scintillae: From cellular automata to computing and cryptography systems," *J. Cellular Automata*, vol. 9, nos. 2–3, pp. 167–181, 2014.

[23] Y. Emek, S. Kutten, R. Lavi, and K. William Moses Jr, "Deterministic leader election in programmable matter," in *Proc. 46th Int. Colloq. Automata, Lang., Program.*, vol. 132, C. Baier, I. Chatzigiannakis, P. Flocchini, S. Leonardi, Eds. Dagstuhl, Germany: Springer, 2019, pp. 1–14.

[24] S. P. Fekete, E. Niehs, C. Scheffer, and A. Schmidt, "Connected reconfiguration of lattice-based cellular structures by finite-memory robots," in *Proc. Int. Symp. Algorithms Exp. Wireless Sensor Netw.*, 2020, pp. 1–5.

[25] P. Flocchini, G. Prencipe, and N. Santoro, *Distributed Computing by Mobile Entities, Current Research in Moving and Computing*, vol. 11340. Cham, Switzerland: Springer, 2019.

[26] N. Gastineau, W. Abdou, N. Mbarek, and O. Togni, "Distributed leader election and computation of local identifiers for programmable matter," in *Proc. 14th Int. Symp. Algorithms Exp. Wireless Sensor Netw.*, vol. 11410, 2019, pp. 159–179.

[27] M. Gauci, E. Monica Ortiz, M. Rubenstein, and R. Nagpal, "Error cascades in collective behavior: A case study of the gradient algorithm on 1000 physical agents," in *Proc. 16th Conf. Auto. Agents MultiAgent Syst.*, Richland, SC, USA, 2017, pp. 1404–1412.

[28] K. Gilpin, A. Knaian, and D. Rus, "Robot pebbles: One centimeter modules for programmable matter through self-disassembly," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2010, pp. 2485–2492.

[29] S. C. Goldstein, J. D. Campbell, and T. C. Mowry, "Programmable matter," *Computer*, vol. 38, no. 6, pp. 99–101, May 2005.

[30] S. C. Goldstein and C. Todd Mowry, "Claytronics: An instance of programmable matter," in *Proc. Wild Crazy Ideas Session ASPLOS*, Boston, MA, USA, Oct. 2004, p. 12.

[31] R. Mayne and A. Adamatzky, *Slime Mould Nanotechnology*. Cham, Switzerland: Springer, 2016, pp. 133–152.

[32] S. Miyashita, S. Guitron, S. Li, and D. Rus, "Robotic metamorphosis by origami exoskeletons," *Sci. Robot.*, vol. 2, no. 10, Sep. 2017, Art. no. eaao4369.

[33] E. Niehs, A. Schmidt, C. Scheffer, D. E. Biediger, M. Yannuzzi, B. Jenett, A. Abdel-Rahman, K. C. Cheung, A. T. Becker, and S. P. Fekete, "Recognition and reconfiguration of lattice-based cellular structures by simple robots," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2020, pp. 8252–8259.

[34] J. Palacci, S. Sacanna, A. P. Steinberg, D. J. Pine, and P. M. Chaikin, "Living crystals of light-activated colloidal surfers," *Science*, vol. 339, no. 6122, pp. 936–940, Feb. 2013.

[35] J. W. Romanishin, K. Gilpin, S. Claici, and D. Rus, "3D M-blocks: Self-reconfiguring robots capable of locomotion via pivoting in three dimensions," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2015, pp. 1925–1932.

[36] M. Sharon, A. S. Rodriguez, C. Sharon, and P. S. Gallardo, *Nanotechnology in the Defense Industry: Advances, Innovation, and Practical Applications*. Hoboken, NJ, USA: Wiley, 2019.

[37] T. Toffoli and N. Margolus, "Programmable matter: Concepts and realization," *Phys. D, Nonlinear Phenomena*, vol. 47, nos. 1–2, pp. 263–272, Jan. 1991.

[38] T. Tucci, B. Piranda, and J. Bourgeois, "A distributed self-assembly planning algorithm for modular robots," in *Proc. 17th Int. Conf. Auto. Agents MultiAgent Syst.*, Richland, SC, USA, 2018, pp. 550–558.

GIANLORENZO D'ANGELO received the B.Sc., M.Sc., and Ph.D. degrees from the University of L'Aquila, Italy, in 2004, 2006, and 2010, respectively.

Before joining GSSI, where he is currently an Associate Professor, he has been a Postdoctoral Researcher with the University of L'Aquila, INRIA Sophia-Antipolis, France, the University of Perugia, Italy, and the University of Pisa, Italy. He is the (co)author of three volumes, six book chapters, and more than 90 research articles in international journals and peer-reviewed conference proceedings. His research interests include design and analysis of algorithms, combinatorial optimization, scheduling, approximation algorithms, network analysis, graph algorithms, distributed computing, and algorithm engineering. He has served as a program committee member of more than 20 scientific events and chaired two international conferences. In 2016, he received the EATCS Award for Best Italian Young Researcher in Theoretical Computer Science.

MATTIA D'EMIDIO received the B.Sc., M.Sc., and Ph.D. degrees from the University of L'Aquila, Italy, in 2008, 2010, and 2014, respectively. He has been an Assistant Professor with the Department of Information Engineering, Computer Science and Mathematics (DISIM), University of L'Aquila, since 2017. Before joining DISIM as an Assistant Professor, he has also been a Visiting Ph.D. Student with INRIA Sophia-Antipolis, since 2013; he held a postdoctoral researcher position at the Department of Computer Science (DI), University of L'Aquila, from 2014 to 2015; he had been a member of the Department of Computer Science, GSSI, as a Postdoctoral Research Fellow, from 2015 to 2017. He is currently a Lecturer and a Scientific Collaborator with the Gran Sasso Science Institute (GSSI), Italy, and a Research Associate with the Istituto Nazionale di Fisica Nucleare (INFN). He is the coauthor of more than 40 research articles that have appeared both in international journals and peer-reviewed conference proceedings, and the coauthor of a book chapter. He has served as both a program committee and an organizing committee member of several international scientific events. His research interests include the design, analysis and efficient implementation of algorithms, distributed systems, and management and processing of massive datasets. His research interests also include algorithm engineering, network optimization and graph algorithms. He received the qualification as an Associate Professor from the Abilitazione Scientifica Nazionale (ASN) Committee of the Italian Ministry of Education, University and Research.

SHANTANU DAS received the Ph.D. degree in computer science from the University of Ottawa, Canada, in 2007. He has been an Associate Professor with Aix-Marseille University, Marseille, France, since 2013. Before that, he has worked at various research institutes, including ETH Zurich, Switzerland, and Technion, Israel, before joining Aix-Marseille University, as a Faculty Member. He is known as an Expert on distributed algorithms and in particular on algorithms for distributed systems of mobile agents or robotic swarms. He has coauthored more than 60 scientific articles and presented seminars and invited lectures at several venues on distributed computing. He was awarded the Chair of excellence in research by CNRS which is the national agency for scientific research in France. The current research interests of Dr. Das include optimization in distributed systems, energy-aware computing, distributed robotics, programmable matter, dynamic networks, and fault tolerance.

ALFREDO NAVARRA received the Ph.D. degree in computer science from the "Sapienza" University of Rome, in 2004. He has been an Associate Professor with the Department of Mathematics and Computer Science, University of Perugia, Italy, since 2015. Before joining the University of Perugia in 2007, he has been with various international research institutes, such as the INRIA of Sophia Antipolis, France; the Department of Computer Science, University of L'Aquila, Italy; and the LaBRI, University of Bordeaux, France. His research interests include algorithms, computational complexity, distributed computing, and networking.

GIUSEPPE PRENCIPE received the Ph.D. degree in computer science from the University of Pisa, Italy, in 2001.

He is currently an Associate Professor with the Department of Computer Science, University of Pisa. His research interests include the study of mobile and distributed systems. In particular, on the design and analysis of algorithms to control and coordinate a set of mobile sensors totally autonomous that can freely move on a plane. Recently, the interest shifted towards the study of Programmable Matter, that refers to a system composed by a great number of small computing (micro and nano level) programmed to collectively solve tasks exclusively using local interactions (neighborhood). Other research interests include the study of mobile agents whose goal is to locate corrupted nodes (called black holes) in a network. Also, some effort was devoted to the study of efficient routing strategies in distributed networks with faulty connections.