Leader Election and Compaction for Asynchronous Silent Programmable Matter

Gianlorenzo D'Angelo
Gran Sasso Science Institute
L'Aquila, Italy
gianlorenzo.dangelo@gssi.it

Mattia D'Emidio
University of L'Aquila
L'Aquila, Italy
mattia.demidio@univaq.it

Shantanu Das
Aix-Marseille University
Marseille, France
shantanu.das@lis-lab.fr

Alfredo Navarra
University of Perugia
Perugia, Italy
alfredo.navarra@unipg.it

Giuseppe Prencipe
University of Pisa
Pisa, Italy
giuseppe.prencipe@unipi.it

ABSTRACT

We study models and algorithms for *Programmable Matter* (PM, shortly), 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. We first introduce SILBOT, a new and weak modeling approach that, unlike previous ones, does not require: i) any synchronization mechanism nor explicit communication between particles; ii) atomicity for the performed actions; iii) activation of one particle at the time within a finite neighborhood. Second, we present a distributed algorithm to solve, in the SILBOT model, a foundational primitive for PM, namely *Leader Election*. This algorithm manages initial configurations that are both *connected* (i.e. particles induce a connected graph) and *compact* (i.e. without *holes*). Third, we show that, if the initial configuration contains holes, it is impossible to achieve leader election while preserving connectivity. Finally, we design an algorithm to handle configurations admitting holes. Specifically, the algorithm achieves *compaction*, i.e. stabilizes the system into a compact connected configuration, while at the same time accomplishing leader election, provided that particles are able to sense holes.

CCS CONCEPTS

• **Theory of computation → Self-organization**; **Distributed algorithms**; • **Computer systems organization → Robotics**;

KEYWORDS

Programmable Matter; Swarm Robotics; Self-Organizing Systems; Leader Election; Compaction; Finite Automata; Distributed Algorithms.

ACM Reference Format:
Gianlorenzo D'Angelo, Mattia D'Emidio, Shantanu Das, Alfredo Navarra, and Giuseppe Prencipe. 2020. Leader Election and Compaction for Asynchronous Silent Programmable Matter. In *Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), Auckland, New Zealand, May 9–13, 2020,* IFAAMAS, 9 pages.

1 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. [32] 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 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., [8, 31]). 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 [25, 26, 30].

The majority of previous works on models for PM have been inspired by physical systems or biological colonies [27, 29]. Among them perhaps the most promising (in terms of quality of the abstraction, as well documented in the literature [10, 18]) is the so-called geometric *Amoebot* model [16], which is inspired by the behavior of 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 [10, 15], bridge building [3], shape formation [6, 16, 18, 33], shape recovery [17], and compression [6]). Essentially all existing solutions, to solve foundational problems in this setting, have resorted to add various features to overcome such hardness. Examples include:

Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), B. An, N. Yorke-Smith, A. El Fallah Seghrouchni, G. Sukthankar (eds.), May 9–13, 2020, Auckland, New Zealand. © 2020 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

- presence of a global serialized synchronizer: no two neighboring particles are assumed to be simultaneously activated (a.k.a. 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.

All 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, 24] 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, 12–14, 21, 23], and metamorphic robots [28].

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 [21] for robotic swarms, extensively studied in distributed computing and widely adopted in practice. Thus, we move a step closer to practically implementable programmable matter. 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.

Within the SILBOT model, we provide distributed algorithms to implement 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 one of the foundational problems in systems of programmable particles [4, 16, 18, 20, 22], 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 [11, 16], for instance, particles resort to randomization to break symmetries; in [4, 11, 16, 22] chirality is assumed, that is a globally consistent circular orientation of the plane shared by all particles; in [20], the scheduler is assumed so as only one particle at the time is activated. Note that, unless randomization [11, 16] or sequential scheduling [20] is assumed, the election of one single leader is not possible. Indeed, in [4] and [18] 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 guide the other particles. We provide a deterministic algorithm for solving leader election when the initial configuration is simply connected as in [18]. Indeed we show how to emulate the erosion process presented in [18] by a simpler set of rules.

When the initial configuration contains *holes*, that is when there is a region of the grid, 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 with arbitrary connected configurations we empower the particles with the capability of detecting whether an empty neighboring cell is part of a hole or it belongs to the exterior grid region. Practically, it is possible for PM particles to acquire such knowledge by local sensing of the surroundings (e.g. via light or pressure sensors) without any global visibility or exchange of messages. We 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. Our algorithms are deterministic, and particles have no additional memory, except the ability to be in two visually distinguishable states.

2 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 modeling assumptions are as follows.

Environment. The system is modeled as an infinite triangular lattice (representing the described hexagonal grid) embedded in the plane, where each node has 6 incident edges. As in the usual model [4, 11, 16, 18, 20, 22], nodes correspond to hexagonal cells and each edge represents a boundary shared by two cells. Each node can contain at most one particle. There are n particles and initially 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. 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. 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 occupancies. 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 may 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, 19]), 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 [10, 18], 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.

Asynchrony and Rounds. The SILBOT model introduces a fine grained notion of asynchrony with possible delays between observations and movements performed by the particles. All operations performed by the particles are not 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.

Orientation and Randomness. We do not make any additional assumption about orientation nor randomness: particles do not agree on clockwise direction (i.e., they have no chirality) and execute deterministic algorithms (i.e., they cannot exploit random outcomes to take decisions). The only random component in the system is of adversarial type [21]: if two contracted particles decide to expand on the same edge simultaneously, exactly one of them succeeds (arbitrarily chosen by the scheduler). 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 and Leader Election. We assume the system is initially in a connected configuration where all particles are CONTRACTED. We define the problem of leader election as in [18] 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 [20]. 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 [18]. 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 2.1 (Particle Leader Election (PLE)). 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.

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 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.

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, resp., from p. 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.

3 PLE WITH SIMPLE CONNECTIVITY

In this section, we provide an 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. We now describe the algorithm called LESC (see the pseudocode in Algorithm 1, it refers to 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 contains all the nodes in this embedding. Informally, our algorithm shrinks 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. Essentially, once expanded, a particle will not compete anymore to become a leader. Formally, 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 [18]). Notice that no contractions are induced by Algorithm 1.

We show that in any deeply simply connected configuration a particle satisfying one of the above conditions always exists until leaders are elected. Moreover, the obtained configuration is guaranteed to be deeply simply connected, while the number of CONTRACTED particles decreases, until it converges to a configuration in Π. In order 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

Algorithm 1: Algorithm LESC (PLE with Simple Connectivity).

```
Precondition  : Each particle is CONTRACTED, the set of nodes
                containing particles induces a simply connected
                subgraph of $G$.
Postcondition : A configuration in $\Pi$.
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 $(p, q)$;
```

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: we call (u, v) 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. We call (u, v) 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 3.1. *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 show the statement of the theorem by proving 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 Fig. 1 (a).
- 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.

The statement then follows as 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. 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. We now show the above three properties separately.

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 is a contradiction, see Fig.s 1 (b) and (c) 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. 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 have shown 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.

We now show that given the existence of such a particle, then there is 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 $|N_1(r)| \leq 2$ (or $|N_1(s)| \leq 2$), then r (or s) will expand, as imposed by Line 3. By referring to Fig.s 2 (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 Fig. 2 (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 Fig. 2 (b)), hence p will expand, which is again a contradiction. Therefore, we have shown 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 Fig. 1 (a). 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 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. By the algorithm, node u was occupied when p has decided to expand, while it is empty when p is CONTRACTED. This implies that the particle that was occupying node u has been CONTRACTED in a time

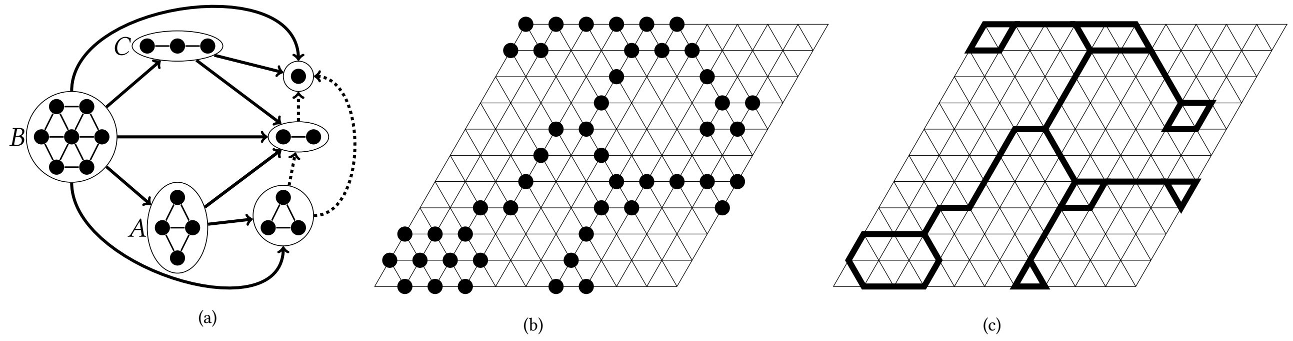

Figure 1: (a) A subgraph of graph H used in the proof of Theorem 3.1. Each vertex is associated with a set of configurations represented by the subgraph induced by only contracted particles. Pending edges are drawn as dashed arrows, while expansion edges are drawn as bold arrows. For simplicity, not all the configurations that according to Algorithm LESC can be obtained from configuration C are depicted. (b) Example of a simply connected configuration. (c) Example of tree of polygons, corresponding to the configuration in (b), used in the proof of Theorem 3.1 for Property P1.

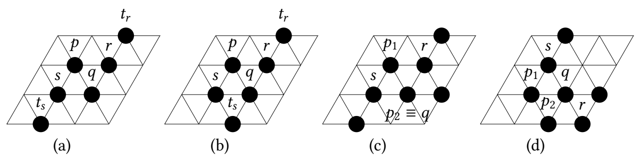

Figure 2: Configurations used in the proof of Theorem 3.1.

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, when it was 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 $|N_1(p)| \leq 3$.

(2) G_t is disconnected. If a particle p is EXPANDED at time t and G_t becomes disconnected because of such 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 case, 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, 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 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 $|N_1(p)| \leq 3$, then we analyze the following 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 Fig. 2 (c)), or p_1 and p_2 share a common central neighbor q and have two further different occupied neighbors, s and r (see Fig. 2 (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.

Property P3. We 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.

Furthermore, 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. □

4 DEALING WITH HOLES

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.

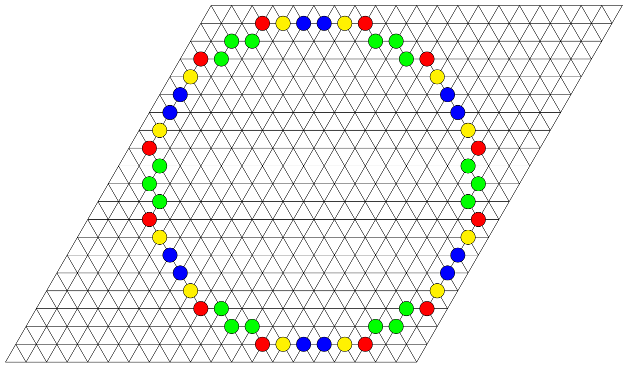

Figure 3: A connected configuration with a hole, used in the proof of Theorem 4.1. Colors identify equivalent particles.

THEOREM 4.1. *Starting from a 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 a connected configuration of particles as shown in Fig. 3. 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 Fig. 3, 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 on local information, may cause the configuration to become disconnected. In the following we assume a particular execution 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., colors) 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.

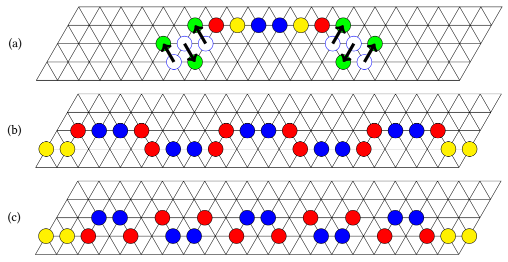

Figure 4: (a) The configuration after one move by green particles. Arrows depict the performed moves. (b) A connected configuration without holes. (c) The configuration obtained after one move by red particles.

Suppose the algorithm makes the blue particles move. We observe any move of a blue particle to any of the four neighboring empty nodes cause the set of particles to be 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 reasoning holds. No matter which of the four neighboring empty nodes is chosen as destination if the scheduler moves all the yellow particles at the same time, then the configuration is disconnected.

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 succession). After such moves, the situation would be as shown in Fig. 4 (a), where a portion of Fig. 3 is reported. Hence, the resulting configuration is disconnected. Thus, the only possibility is that the algorithm instructs all red particles to move first to the neighboring nodes that are inside the hole (see Fig. 3); 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 Fig. 4 (b). Note that the red particles in this configuration have the same view as the red particles in the configuration from Fig. 3. 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 as shown in Fig. 4 (c) is one where the set of particles is disconnected. Hence it is impossible to solve PLE without disconnecting the set of particles. □

Note that it is not straightforward to re-establish a connected configuration once the particles are disconnected. 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. Clearly, if the configuration remains disconnected, there may be too many leaders elected. On the one hand, it is clear that Theorem 4.1 does not prove the impossibility of solving PLE once the configuration disconnects. On the other hand we observe it is of practical interest to maintain connectivity. To this end, we empower 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, the capability is defined as follows:

Definition 4.2 (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 show that particles with exterior awareness are able to reduce any connected configuration to a simply connected one 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 4.1.

5 COMPACTION AND ELECTION

In this section, we assume particles are endowed with the Exterior Awareness capability and show how, starting from any arbitrary connected configuration (hence not necessarily simply, possibly containing holes), we can obtain a deeply simply connected configuration, i.e., we provide an algorithm to achieve *compaction of holes*. We then show how this process also leads to elect a set of at most three leaders as Algorithm LESC.

Our algorithm achieving compaction and leader election is named Algorithm CHLE and its pseudocode is provided in Algorithm 2, again from the point of view of a single particle. To describe it, we use the following further notation: given a particle p, we denote by $N_1(p,\textsc{c})$ and $N_1(p,\textsc{i})$ the set of CONT and IN nodes adjacent to p, resp., and by $N_1(p,\textsc{ic})$ the set $\{N_1(p,\textsc{c}) \cup N_1(p,\textsc{i})\}$.

The rationale of Algorithm CHLE is similar to that of Algorithm LESC, that is we aim at shrinking 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, with the difference that now we take into account IN nodes in the envelope. In 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, similarly to Algorithm LESC, Algorithm CHLE allows to expand only particles p such that $G(N_1(p,\textsc{ic}))$ is connected and $|N_1(p,\textsc{ic})| \leq 3$. The main difference is that, when a node in $N_1(p,\textsc{ic})$ is an empty internal node (i.e.$|N_1(p,\textsc{i})| = 1$) 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 $|N_1(p,\textsc{ic})| = |N_1(p,\textsc{i})| = 1$, then p expands toward the only internal node, while if $|N_1(p,\textsc{ic})| = |N_1(p,\textsc{c})| = 1$ it expands toward the only contracted particle (see Line 3). If $|N_1(p,\textsc{ic})| = 2$ and $G(N_1(p,\textsc{ic}))$ is connected, instead, then: i) if there is a node q in $N_1(p,\textsc{i})$, p expands along edge (p,q) (Line 5); ii) otherwise, it expands toward any contracted particle (Line 6). Finally, if $|N_1(p,\textsc{ic})| = 3$ and $G(N_1(p,\textsc{ic}))$ is connected, then we have two cases: either there is one empty internal node or all the three neighbors in $N_1(p,\textsc{ic})$ are occupied by contracted particles. In the former case, p expands toward the internal node (Line 10), otherwise it expands along the central contracted neighbor only if the condition at Line 11 is satisfied. Note that a particle p with connected $G(N_1(p,\textsc{ic}))$, $|N_1(p,\textsc{ic})| = 3$, and $|N_1(p,\textsc{i})| \geq 2$ cannot exist.

Notice that, unlike 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. 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.

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 3.1 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.

Algorithm 2: Algorithm CHLE (Compaction of Holes and PLE)

```
Precondition  : A connected configuration where each particle is
                CONTRACTED.
Postcondition : A configuration in $\Pi$.
1  if ($p$ is CONTRACTED) $\wedge$ ($G(N_1(p,\textsc{ic}))$ is connected) $\wedge$
     ($G(N_1(p,\textsc{c}) \cup N_2(p) \cup \{p\}) \notin \Pi$) 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 Expand along $(p,q)$;
6          else Expand along $(p,r)$;
7      if $N_1(p,IC) = \{r,q,s\}$ then
8          if $\{r,q,s\} \cap N_1(p,I) \neq \emptyset$ then
9              Let $x$ be the only element of $\{r,q,s\} \cap N_1(p,\textsc{i})$;
10             Expand along $(p,x)$;
11         else if $|N_1(r,c)| > 2 \wedge |N_1(q,c)| > 3 \wedge |N_1(s,c)| > 2$
             then Expand along $(p,q)$;
```

THEOREM 5.1. *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.*

SKETCH OF PROOF. Since from simply connected configurations Algorithm CHLE behaves like Algorithm LESC, we can assume that, in the initial configuration, n CONTRACTED particles form a non-simply connected configuration, i.e. there is at least one IN node.

To prove the statement, we use an argument similar to that used in the proof of Theorem 3.1, that is we model an execution of Algorithm CHLE as a path in a directed graph $H = (V,E)$. However, in this case, vertices of the graph represent connected configurations (including semi-occupied nodes) with n CONTRACTED or EXPANDED particles, and edges correspond to expansions or contractions induced by Algorithm CHLE. More precisely, each vertex $u \in V$ corresponds to a connected configuration C_u, and there is a directed edge $(u,v) \in E$ if there exists an execution schedule of Algorithm CHLE that leads from C_u to C_v, without generating in between further configurations different from C_v.
We can show the 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 in Fig. 1 (a).

P2. Each expansion or contraction dictated by Algorithm CHLE corresponds to an edge in H, i.e., the configuration (including semi-occupied nodes) obtained by any expansion or contraction of CHLE is connected.

P3. Graph H is acyclic.

The above claims can be proven separately and by Properties P1, P2 and P3 it follows that the algorithm terminates in a configuration in Π, thus achieving compaction and leader election. Moreover, each path in H starting from the vertex corresponding to the initial configuration C and ending in a vertex corresponding to a configuration in Π has a length bounded by the sum of the number of contracted particles and the number of IN nodes in C. As the former is n and the latter is bounded by n^2, the statement follows. □

6 CONCLUSIONS AND EXTENSIONS

In this paper, we have proposed the SILBOT model, a new, weaker, and more realistic modeling approach for programmable matter, where particles act asynchronously and independently, are silent (no direct communication) and rely on just one bit of persistent memory. Despite the weak computational power of this setting, we have shown, by designing a corresponding algorithm, that the leader election problem, one of the foundational tasks for PM systems, can be solved if the initial configuration is simply connected. Furthermore, we have also proven that it is impossible to solve leader election while preserving connectivity, if the initial configuration contains holes. Finally, we have shown, again by giving an appropriate algorithm, how endowing particles with the capability of distinguishing exterior cells from interior cells of the grid suffices to achieve both compaction of holes and leader election.

As future work, we aim at investigating which tasks, besides leader election, can be successfully performed under SILBOT and which cannot. Also, in the case of impossibility results it would be worth studying what are the minimal capabilities that must be added to the PM systems to achieve feasibility (e.g. few more states or other very simple abilities like optical signalling [9, 12]). An interesting research direction might be that of designing a simulation environment where to test and compare existing models, and where to handle new rules or capabilities for the particles.

7 ACKNOWLEDGEMENTS

The work has been partially supported by the European project "Geospatial based Environment for Optimisation Systems Addressing Fire Emergencies" (GEO-SAFE) - contract no. H2020-691161, by the Italian MIUR PRIN 2017 Project ALGADIMAR "Algorithms, Games, and Digital Markets", by the Italian National Group for Scientific Computation GNCS-INdAM, and by Progetto PRA_2018_43 (Università di Pisa, Italy).

REFERENCES

[1] Andrew Adamatzky (Ed.). 2010. *Game of Life Cellular Automata*. Springer.

[2] Michael Amir and Alfred M Bruckstein. 2019. Minimizing Travel in the Uniform Dispersal Problem for Robotic Sensors. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS '19)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 113–121.

[3] Marta Andrés Arroyo, Sarah Cannon, Joshua J. Daymude, Dana Randall, and Andréa W. Richa. 2018. A stochastic approach to shortcut bridging in programmable matter. *Natural Computing* 17, 4 (2018), 723–741.

[4] Rida A. Bazzi and Joseph L. Briones. 2019. Deterministic Leader Election in Self-organizing Particle Systems. In *Proceedings of the 21st International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS)*, Vol. 11914. Springer.

[5] Aaron Becker, Yan Ou, Paul Kim, Min Jun Kim, and Agung Julius. 2013. Feedback control of many magnetized: Tetrahymena pyriformis cells by exploiting phase inhomogeneity. In *Proc. of the International Conf. on Intelligent Robots and Systems*. IEEE, 3317–3323.

[6] Sarah Cannon, Joshua J. Daymude, Dana Randall, and Andréa W. Richa. 2016. A Markov Chain Algorithm for Compression in Self-Organizing Particle Systems. In *Proceedings of the 35th Symp. on Principles of Distributed Computing (PODC)*. 279–288.

[7] Sarah Cannon, Joshua J. Daymude, William Savoie, Ross Warkentin, Shengkai Li, Daniel I. Goldman, Dana Randall, and Andréa W. Richa. 2017. Phototactic Supersmarticles. *CoRR* abs/1711.01327 (2017).

[8] Sonia Contera. 2019. *Nano Comes to Life: How Nanotechnology Is Transforming Medicine and the Future of Biology*. Princeton University Press.

[9] Shantanu Das, Paola Flocchini, Giuseppe Prencipe, Nicola Santoro, and Masafumi Yamashita. 2016. Autonomous Mobile Robots with Lights. *Theor. Comput. Sci.* 609 (2016), 171–184. https://doi.org/10.1016/j.tcs.2015.09.018

[10] Joshua J. Daymude, Zahra Derakhshandeh, Robert Gmyr, Alexandra Porter, Andréa W. Richa, Christian Scheideler, and Thim Strothmann. 2018. On the runtime of universal coating for programmable matter. *Natural Computing* 17, 1 (2018), 81–96.

[11] Joshua J. Daymude, Robert Gmyr, Andréa W. Richa, Christian Scheideler, and Thim Strothmann. 2017. Improved Leader Election for Self-organizing Programmable Matter. In *Proceedings of the 13th International Symp. on Algorithms and Experiments for Wireless Sensor Networks (ALGOSENSORS)*, Vol. 10718. Springer, 127–140.

[12] Mattia D'Emidio, Gabriele Di Stefano, Daniele Frigioni, and Alfredo Navarra. 2018. Characterizing the computational power of mobile robots on graphs and implications for the Euclidean plane. *Inf. Comput.* 263 (2018), 57–74.

[13] Mattia D'Emidio, Daniele Frigioni, and Alfredo Navarra. 2016. Characterizing the Computational Power of Anonymous Mobile Robots. In *36th IEEE International Conference on Distributed Computing Systems, ICDCS 2016, Nara, Japan, June 27-30, 2016*. IEEE Computer Society, 293–302.

[14] Mattia D'Emidio, Daniele Frigioni, and Alfredo Navarra. 2016. Synchronous Robots vs Asynchronous Lights-Enhanced Robots on Graphs. *Electr. Notes Theor. Comput. Sci.* 322 (2016), 169–180.

[15] Zahra Derakhshandeh, Robert Gmyr, Andréa W. Richa, Christian Scheideler, and Thim Strothmann. 2017. Universal coating for programmable matter. *Theor. Comput. Sci.* 671 (2017), 56–68.

[16] Zahra Derakhshandeh, Robert Gmyr, Thim Strothmann, Rida A. Bazzi, Andréa W. Richa, and Christian Scheideler. 2015. Leader Election and Shape Formation with Self-organizing Programmable Matter. In *Proceedings of 21st International Conf. on DNA Computing and Molecular Programming (DNA)*, Vol. 9211. Springer, 117–132.

[17] Giuseppe Antonio Di Luna, Paola Flocchini, Giuseppe Prencipe, Nicola Santoro, and Giovanni Viglietta. 2018. Line Recovery by Programmable Particles. In *Proceedings of the 19th International Conf. on Distributed Computing and Networking, (ICDCN)*. ACM, 4:1–4:10.

[18] Giuseppe A. Di Luna, Paola Flocchini, Nicola Santoro, Giovanni Viglietta, and Yukiko Yamauchi. 2020. Shape Formation by Programmable Particles. *Distributed Computing* 33, 1 (2020), 69–101.

[19] Gabriele Di Stefano and Alfredo Navarra. 2014. The Game of Scintillae: From Cellular Automata to Computing and Cryptography Systems. *J. Cellular Automata* 9, 2-3 (2014), 167–181.

[20] Yuval Emek, Shay Kutten, Ron Lavi, and William K. Moses Jr. 2019. Deterministic Leader Election in Programmable Matter. In *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019) (Leibniz International Proceedings in Informatics (LIPIcs))*, Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi (Eds.), Vol. 132. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 140:1–140:14. https://doi.org/10.4230/LIPIcs.ICALP.2019.140

[21] Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro (Eds.). 2019. *Distributed Computing by Mobile Entities, Current Research in Moving and Computing*. Vol. 11340. Springer.

[22] Nicolas Gastineau, Wahabou Abdou, Nader Mbarek, and Olivier Togni. 2019. Distributed Leader Election and Computation of Local Identifiers for Programmable Matter. In *Proceedings of the 14th International Symp. on Algorithms and Experiments for Wireless Sensor Networks (ALGOSENSORS)*, Vol. 11410. Springer, 159–179.

[23] Melvin Gauci, Monica E. Ortiz, Michael Rubenstein, and Radhika Nagpal. 2017. Error Cascades in Collective Behavior: A Case Study of the Gradient Algorithm on 1000 Physical Agents. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems (AAMAS '17)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 1404–1412.

[24] Kyle Gilpin, Ara Knaian, and Daniela Rus. 2010. Robot pebbles: One centimeter modules for programmable matter through self-disassembly. In *2010 IEEE International Conference on Robotics and Automation*. IEEE, IEEE, 2485–2492.

[25] Seth Copen Goldstein, Jason Campbell, and Todd C. Mowry. 2005. Programmable Matter. *IEEE Computer* 38, 6 (2005), 99–101.

[26] Seth Copen Goldstein and Todd C. Mowry. 2004. Claytronics: An Instance of Programmable Matter. In *Wild and Crazy Ideas Session of ASPLOS*. Boston, MA.

[27] Richard Mayne and Andrew Adamatzky. 2016. *Slime Mould Nanotechnology*. Springer International Publishing, Cham, 133–152. https://doi.org/10.1007/978-3-319-26662-6_7

[28] Shuhei Miyashita, Steven Guitron, Shuguang Li, and Daniela Rus. 2017. Robotic metamorphosis by origami exoskeletons. *Science Robotics* 2, 10 (2017).

[29] Jeremie Palacci, Stefano Sacanna, Asher Preska Steinberg, David J. Pine, and Paul M. Chaikin. 2013. Living Crystals of Light-Activated Colloidal Surfers. *Science* 339, 6122 (2013), 936–940.

[30] John W Romanishin, Kyle Gilpin, Sebastian Claici, and Daniela Rus. 2015. 3D M-Blocks: Self-reconfiguring robots capable of locomotion via pivoting in three dimensions. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, IEEE, 1925–1932.

[31] Madhuri Sharon, Angelica SL Rodriguez, Chetna Sharon, and Pio Sifuentes Gallardo. 2019. *Nanotechnology in the Defense Industry: Advances, Innovation, and Practical Applications*. John Wiley & Sons.

[32] Tommaso Toffoli and Norman Margolus. 1991. Programmable matter: Concepts and realization. *Physica D: Nonlinear Phenomena* 47, 1 (1991), 263 – 272.

[33] Thadeu Tucci, Benoundefinedt Piranda, and Julien Bourgeois. 2018. A Distributed Self-Assembly Planning Algorithm for Modular Robots. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS '18)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 550–558.