CrossMark

On the runtime of universal coating for programmable matter

Joshua J. Daymude[1] · Zahra Derakhshandeh[1] · Robert Gmyr[2] · Alexandra Porter[1] · Andréa W. Richa[1] · Christian Scheideler[2] · Thim Strothmann[2]

Abstract

Imagine coating buildings and bridges with smart particles (also coined smart paint) that monitor structural integrity and sense and report on traffic and wind loads, leading to technology that could do such inspection jobs faster and cheaper and increase safety at the same time. In this paper, we study the problem of uniformly coating objects of arbitrary shape in the context of *self-organizing programmable matter*, i.e., programmable matter which consists of simple computational elements called particles that can establish and release bonds and can actively move in a self-organized way. Particles are anonymous, have constant-size memory, and utilize only local interactions in order to coat an object. We continue the study of our universal coating algorithm by focusing on its runtime analysis, showing that our algorithm terminates within a *linear number of rounds* with high probability. We also present a matching linear lower bound that holds with high probability. We use this lower bound to show a *linear lower bound on the competitive gap* between fully local coating algorithms and coating algorithms that rely on global information, which implies that our algorithm is also optimal in a competitive sense. Simulation results show that the competitive ratio of our algorithm may be better than linear in practice.

Keywords Distributed algorithms · Programmable matter · Self-organization · Self-organizing systems · Coating

1 Introduction

Inspection of bridges, tunnels, wind turbines, and other large civil engineering structures for defects is a time-consuming, costly, and potentially dangerous task. In the future, *smart coating* technology, or *smart paint*, could do the job more efficiently and without putting people in danger. The idea behind smart coating is to form a thin layer of a specific substance on an object which then makes it possible to measure a condition of the surface (such as temperature or cracks) at any location, without direct access to the location. The concept of smart coating already occurs in nature, such as proteins closing wounds, antibodies surrounding bacteria, or ants surrounding food to transport it to their nest. These diverse examples suggest a broad range of applications of smart coating technology in the future, including repairing cracks or monitoring tension on bridges, repairing space craft, fixing leaks in a nuclear reactor, or stopping internal bleeding. We continue the study of coating problems in the context of self-organizing

✉ Joshua J. Daymude
jdaymude@asu.edu

Zahra Derakhshandeh
zderakhs@asu.edu

Robert Gmyr
gmyr@mail.upb.de

Alexandra Porter
amporte6@asu.edu

Andréa W. Richa
aricha@asu.edu

Christian Scheideler
scheidel@mail.upb.de

Thim Strothmann
thim@mail.upb.de

1 Computer Science, CIDSE, Arizona State University, Tempe, AZ, USA

2 Department of Computer Science, Paderborn University, Paderborn, Germany

programmable matter consisting of simple computational elements, called particles, that can establish and release bonds and can actively move in a self-organized way using the geometric version of the amoebot model presented in Daymude et al. (2015), Derakhshandeh et al. (2014). In doing so, we proceed to investigate the runtime analysis of our universal coating algorithm, introduced in Derakhshandeh et al. (2017). We first show that coating problems do not only have a (trivial) linear lower bound on the runtime, but that there is also a linear lower bound on the competitive gap between the runtime of fully local coating algorithms and coating algorithms that rely on global information. We then investigate the worst-case time complexity of our universal coating algorithm and show that it terminates within a linear number of rounds with high probability (w.h.p.),[1] which implies that our algorithm is optimal in terms of worst-case runtime and also in a competitive sense. Moreover, our simulation results show that in practice the competitive ratio of our algorithm is often better than linear.

1.1 Amoebot model

In the *amoebot model*, space is modeled as an infinite, undirected graph G whose vertices are positions that can be occupied by at most one particle and whose edges represent all possible atomic transitions between these positions. In the *geometric amoebot model*, we further assume that $G = G_{eqt}$, where $G_{eqt} = (V_{eqt}, E_{eqt})$ is the infinite regular triangular grid graph (see Fig. 1a). Each particle occupies either a single node (i.e., it is *contracted*) or a pair of adjacent nodes in G_{eqt} (i.e., it is *expanded*), as in Fig. 1b. Particles move by executing a series of *expansions* and *contractions*: a contracted particle can expand into an unoccupied adjacent node to become expanded, and completes its movement by contracting to once again occupy a single node. For an expanded particle, we denote the node it last expanded into as its *head* and the other node it occupies as its *tail*; for a contracted particle, the single node it occupies is both its head and its tail.

Two particles occupying adjacent nodes are said to be *neighbors* and are connected by a *bond*. These bonds both ensure that the overall particle system remains connected as well as providing a mechanism for exchanging information between particles. In order to maintain connectivity as they move, neighboring particles coordinate their motion in a *handover*, which can occur in two ways. A contracted particle p can initiate a handover by expanding into a node occupied by an expanded neighbor q, "pushing" q and
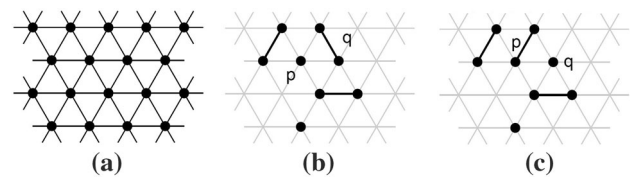
Fig. 1 **a** Shows a section of G_{eqt}, where nodes of G_{eqt} are shown as black circles. **b** Shows five particles on G_{eqt}; the underlying graph G_{eqt} is depicted as a gray mesh; a contracted particle is depicted as a single black circle and an expanded particle is depicted as two black circles connected by an edge. **c** Depicts the resulting configuration after a handover was performed by particles p and q in (**b**)

forcing it to contract. Alternatively, an expanded particle q can initiate a handover by contracting, "pulling" a contracted neighbor p to the node it is vacating, thereby forcing p to expand. Figure 1b, c illustrates two particles performing a handover.

Particles are *anonymous*, but each keeps a collection of uniquely labeled *ports* corresponding to the edges incident to the node(s) it occupies. Bonds between neighboring particles are formed through ports that face each other. The particles are assumed to have a common *chirality*, meaning they share the same notion of *clockwise (CW) direction*. This allows each particle to label its ports counting in the clockwise direction; without loss of generality, we assume each particle labels its head and tail ports from 0 to 5. However, particles may have different offsets for their port labels, and thus do not share a common sense of orientation. Each particle has a constant-size, local memory for which both it and its neighbors have read and write access. Particles can communicate by writing into each other's memories. Due to the limitation of constant-size memory, particles have no knowledge of the total number of particles in the system, nor do they have any approximation of this value. We assume that any conflicts of movement or simultaneous memory writes are resolved arbitrarily, so that at most one particle writes to any memory location or moves into an empty position at any given time.

The *configuration C* of the particle system at the beginning of time t consists of (1) the nodes in G_{eqt} occupied by the object and the set of particles, and (2) the current state of each particle, including whether it is expanded or contracted, its port labeling, and the contents of its local memory.

Following the standard asynchronous model of computation (Lynch 1996), we assume that the system progresses through a sequence of atomic *activations* of individual particles. When activated, a particle can perform a bounded amount of computation involving its local memory and the memories of its neighbors and at most one movement. A classical result under this model is that for any asynchronous concurrent execution of atomic activations, there exists a sequential ordering of the activations which

---

produces the same end configuration, provided conflicts arising from the concurrent execution are resolved (as they are in our scenario). We assume the resulting activation sequence is *fair*,[2] i.e., for each particle p and any time t, p will eventually be activated at some time $t' > t$. An *asynchronous round* is complete once every particle has been activated at least once.

1.2 Universal coating problem

In the *universal coating problem* we consider an instance (P, O) where P represents the particle system and O represents the fixed object to be coated. Let $n = |P|$ be the number of particles in the system, $V(P)$ be the set of nodes occupied by P, and $V(O)$ be the set of nodes occupied by O (when clear from the context, we may omit the $V(\cdot)$ notation). For any two nodes $v, w \in V_{eqt}$, the *distance* $d(v, w)$ between v and w is the length of the shortest path in G_{eqt} from v to w. The distance $d(v, U)$ between a $v \in V_{eqt}$ and $U \subseteq V_{eqt}$ is defined as $\min_{w \in U} d(v, w)$. Define *layer i* to be the set of nodes that have a distance i to the object, and let B_i be the number of nodes in layer i. An instance is *valid* if the following properties hold:

1. The particles are all contracted and are initially in the *idle* state.
2. The subgraphs of G_{eqt} induced by $V(O)$ and $V(P) \cup V(O)$, respectively, are connected, i.e., there is a single object and the particle system is connected to the object.
3. The subgraph of G_{eqt} induced by $V_{eqt} \setminus V(O)$ is connected, i.e., the object O has no holes.[3]
4. $V_{eqt} \setminus V(O)$ is $2(\lceil \frac{n}{B_1} \rceil + 1)$-connected, i.e., O cannot form *tunnels* of width less than $2(\lceil \frac{n}{B_1} \rceil + 1)$.

Note that a width of at least $2\lceil \frac{n}{B_1} \rceil$ is needed to guarantee that the object can be evenly coated. The coating of narrow tunnels requires specific technical mechanisms that complicate the protocol without contributing to the basic idea of coating, so we ignore such cases in favor of simplicity.

A configuration C is *legal* if and only if all particles are contracted and

$$\min_{v \in V_{eqt} \setminus (V(P) \cup V(O))} d(v, V(O)) \geq \max_{v \in V(P)} d(v, V(O)),$$

meaning that all particles are as close to the object as possible or *coat O as evenly as possible*. A configuration C is said to be *stable* if no particle in C ever performs a state change or

movement. An algorithm *solves* the universal coating problem if, starting from any valid instance, it reaches a *stable legal configuration* in a finite number of rounds.

1.3 Related work

Many approaches have been proposed with potential applications in smart coating; these can be categorized as active and passive systems. In passive systems, particles move based only on their structural properties and interactions with their environment, or have only limited computational ability but lack control of their motion. Examples include population protocols (Angluin et al. 2006) as well as molecular computing models such as DNA self-assembly systems (see, e.g., the surveys in Doty 2012; Patitz 2014; Woods 2015) and slime molds (Bonifaci et al. 2012; Li et al. 2010).

Our focus, however, is on active systems, in which computational particles control their actions and motions to complete specific tasks. Coating has been extensively studied in the area of *swarm robotics*, but not commonly treated as a stand-alone problem; it is instead examined as part of *collective transport* (e.g., Wilson et al. 2014) or *collective perception* (e.g., see respective section of Brambilla et al. 2013). Some research focuses on coating objects as an independent task under the name of *target surrounding* or *boundary coverage*. The techniques used in this context include stochastic robot behaviors (Kumar and Berman 2014; Pavlic et al. 2016), rule-based control mechanisms Blázovics et al. (2012a) and potential field-based approaches (Blázovics et al. 2012b). While the analytic techniques developed in swarm robotics are somewhat relevant to this work, many such systems assume more computational power and movement capabilities than the model studied in this work does. Michail and Spirakis recently proposed a model (Michail and Spirakis 2016) for network construction inspired by population protocols (Angluin et al. 2006). The population protocol model is related to self-organizing particle systems, but is different in that agents (corresponding to our particles) can move freely in space and establish connections at any time. It would, however, be possible to adapt their approach to study coating problems under the population protocol model.

In the context of molecular programming, our model most closely relates to the *nubot* model by Woods et al. (2013), Chen et al. (2015), which seeks to provide a framework for rigorous algorithmic research on self-assembly systems composed of active molecular components, emphasizing the interactions between molecular structure and active dynamics. This model shares many characteristics of our amoebot model (e.g., space is modeled as a triangular grid, nubot monomers have limited computational abilities, and there is no global orientation)

---

[2] We will see this notion of fairness is sufficient to prove the desired runtime for our algorithm; no further assumptions regarding the distribution of the activation sequence are necessary.

[3] If O does contain holes, we consider the subset of particles in each connected region of $V_{eqt} \setminus V(O)$ separately.

but differs in that nubot monomers can replicate or die and can perform coordinated rigid body movements. These additional capabilities prohibit the direct translation of results under the nubot model to our amoebot model; the latter provides a framework for future, large-scale swarm robotic systems of computationally limited particles (each possibly at the nano- or micro-scale) with only local control and coordination mechanisms, where these capabilities would likely not apply.

Finally, in Derakhshandeh et al. (2017) we presented our Universal Coating algorithm and proved its correctness. We also showed it to be worst-case work-optimal, where work is measured in terms of number of particle movements.

1.4 Our contributions

In this paper we continue the analysis of the *Universal Coating algorithm* introduced in Derakhshandeh et al. (2017). As our main contribution in this paper, we investigate the runtime of our algorithm and prove that our algorithm terminates within a *linear number of rounds* with high probability. This result relies, in part, on an update to the leader election protocol used in Derakhshandeh et al. (2017) which is fully defined and analyzed in Daymude et al. (2017). We also present a matching linear lower bound for any *local-control* coating algorithm (i.e., one which uses only local information in its execution) that holds with high probability. We use this lower bound to show a *linear lower bound on the competitive gap* between fully local coating algorithms and coating algorithms that rely on global information, which implies that our algorithm is also optimal in a competitive sense. We then present some simulation results demonstrating that in practice the competitive ratio of our algorithm is often much better than linear.

1.4.1 Overview

In Sect. 2, we again present the algorithm introduced in Derakhshandeh et al. (2017). We then present a comprehensive formal runtime analysis of our algorithm, by first presenting some lower bounds on the competitive ratio of any local-control algorithm in Sect. 3, and then proving that our algorithm has a runtime of $\mathcal{O}(n)$ rounds w.h.p. in Sect. 4, which matches our lower bounds.

2 Universal coating algorithm

In this section, we summarize the universal coating algorithm introduced in Derakhshandeh et al. (2017) (see Derakhshandeh et al. 2017 for a detailed description). This algorithm is constructed by combining a number of asynchronous primitives, which are integrated seamlessly without any underlying synchronization. The *spanning forest* primitive organizes the particles into a spanning forest, which determines the movement of particles while preserving system connectivity; the *complaint-based coating* primitive coats the first layer by bringing any particles not yet touching the object into the first layer while there is still room; the *general layering* primitive allows each layer i to form only after layer $i - 1$ has been completed, for $i \geq 2$; and the *node-based leader election* primitive elects a node in layer 1 whose occupant becomes the leader particle, which is used to trigger the general layering process for higher layers.

2.1 Preliminaries

We define the set of *states* that a particle can be in as *idle*, *follower*, *root*, and *retired*. In addition to its state, a particle maintains a constant number of other flags, which in our context are constant size pieces of information visible to neighboring particles. A flag x owned by some particle p is denoted by $p.x$. Recall that a *layer* is the set of nodes $V \subseteq V_{\text{eqt}}$ that are equidistant to the object O. A particle keeps track of its current layer number in $p.layer$. In order to respect the constant-size memory constraint of particles, we take all layer numbers modulo 4. Each root particle p has a flag $p.down$ which stores a port label pointing to a node of the object if $p.layer = 1$, and to an occupied node adjacent to its head in layer $p.layer - 1$ if $p.layer > 1$. We now describe the coating primitives in more detail.

2.2 Coating primitives

The *spanning forest primitive* (Algorithm 1) organizes the particles into a spanning forest $\mathcal{F}$, which yields a straightforward mechanism for particles to move while preserving connectivity (see Daymude et al. 2015; Derakhshandeh et al. 2015 for details). Initially, all particles are *idle*. A particle p touching the object changes its state to *root*. For any other idle particle p, if p has a root or a follower in its neighborhood, it stores the direction to one of them in *p.parent*, changes its state to *follower*, and generates a complaint flag; otherwise, it remains idle. A follower particle p uses handovers to follow its parent and updates the direction *p.parent* as it moves in order to maintain the same parent in the tree (note that the particular particle at *p.parent* may change due to p's parent performing a handover with another of its children). In this way, the trees formed by the parent relations stay connected, occupy only the nodes they covered before, and do not mix with other trees. A root particle p uses the flag *p.dir* to determine its movement direction. As p moves, it

updates *p.dir* so that it always points to the next position of a clockwise movement around the object. For any particle *p*, we call the particle occupying the position that *p.parent* resp. *p.dir* points to the *predecessor* of *p*. If a root particle does not have a predecessor, we call it a *super-root*.

contracted super-root *p* only expands to *p.dir* if it holds at least one complaint flag, and when it expands, it consumes one of these complaint flags. All other roots *p* move towards *p.dir* whenever possible (i.e., no complaint flags are required) by performing a handover with

---

Algorithm 1 Spanning Forest Primitive

A particle *p* acts depending on its state as described below:

idle: If *p* is adjacent to the object *O*, it becomes a *root* particle, makes the current node it occupies a *leader candidate position*, and starts running the leader election algorithm. If *p* is adjacent to a *retired* particle, *p* also becomes a *root* particle. If a neighbor *p'* is a root or a follower, *p* sets the flag *p.parent* to the label of the port to *p'*, puts a *complaint flag* in its local memory, and becomes a *follower*. If none of the above applies, *p* remains idle.

follower: If *p* is contracted and adjacent to a retired particle or to *O*, then *p* becomes a *root* particle. If *p* is contracted and has an expanded parent, then *p* initiates HANDOVER(*p*) (Algorithm 2); otherwise, if *p* is expanded, it considers the following two cases: (*i*) if *p* has a contracted child particle *q*, then *p* initiates HANDOVER(*p*); (*ii*) if *p* has no children and no idle neighbor, then *p* contracts. Finally, if *p* is contracted, it runs the function FORWARDCOMPLAINT(*p*, *p.parent*) (Algorithm 3).

root: If particle *p* is in layer 1, *p* participates in the leader election process. If *p* is contracted, it first executes MARKERRETIREDCONDITIONS(*p*) (Algorithm 6) and becomes *retired*, and possibly also a *marker*, accordingly. If *p* does not become retired, then if it has an expanded root in *p.dir*, it initiates HANDOVER(*p*); otherwise, *p* calls LAYEREXTENSION(*p*) (Algorithm 4). If *p* is expanded, it considers the following two cases: (*i*) if *p* has a contracted child, then *p* initiates HANDOVER(*p*); (*ii*) if *p* has no children and no idle neighbor, then *p* contracts. Finally, if *p* is contracted, it runs FORWARDCOMPLAINT(*p*, *p.dir*).

retired: *p* clears a potential complaint flag from its memory and performs no further action.

The *complaint-based coating primitive* is used for the coating of the first layer. Each time a particle *p* holding at least one complaint flag is activated, it forwards one to its predecessor as long as that predecessor holds less than two complaint flags. We allow each particle to hold up to two complaint flags to ensure that a constant size memory is sufficient for storing the complaint flags and so the flags quickly move forward to the super-roots. A

their predecessor (which must be another root) or a successor (which is a root or follower of its tree), with preference given to a follower so that additional particles enter layer 1. As we will see, these rules ensure that whenever there are particles in the system that are not yet at layer 1, eventually one of these particles will move to layer 1, unless layer 1 is already completely filled with contracted particles.

Algorithm 2 HANDOVER(*p*)

1: **if** *p* is expanded **then**
2:     **if** *p.layer* = 1 and *p* has a follower child *q* such that *q.parent* points to the tail of *p* **then**
3:         **if** *q* is contracted **then**
4:             *p* initiates a handover with particle *q*
5:     **else if** *p* has a contracted (follower or root) child *q* such that *q.parent* points to the tail of *p* **then**
6:         *p* initiates a handover with particle *q*
7: **else if** *p* has an expanded parent *q* or the position in *p.dir* is occupied by an expanded root *q* **then**
8:     *p* initiates a handover with particle *q*

Algorithm 3 FORWARDCOMPLAINT(p, i)

1: **if** p holds at least one complaint flag **and** the particle q adjacent to p in direction
   i holds less than two complaint flags **then**
2:     p forwards one complaint flag to q

The *leader election primitive* runs during the complaint-based coating primitive to elect a node in layer 1 as the leader position. This primitive is similar to the algorithm presented in Daymude et al. (2015) with the difference that leader candidates are nodes instead of static particles (which is important because in our case particles may still move during the leader election primitive). The primitive only terminates once all positions in layer 1 are occupied. Once the leader position is determined, all positions in layer 1 are filled by contracted particles and whatever particle currently occupies that position becomes the *leader*. This leader becomes a marker particle, marking a neighboring position in the next layer as a *marked position* which determines a starting point for layer 2, and becomes *retired*. Once a contracted root p has a retired particle in the direction *p.dir*, it retires as well, which causes the particles in layer 1 to become retired in counter-clockwise order. At this point, the general layering primitive becomes active, which builds subsequent layers until there are no longer followers in the system. If the leader election primitive does not terminate (which only happens if $n < B_1$ and layer 1 is never completely filled), then the complaint flags ensure that the super-roots eventually stop, which eventually results in a stable legal coating.

Algorithm 4 LAYEREXTENSION(p)

    Calculating $p.layer$, $p.down$ **and** $p.dir$
1: The layer number of any node occupied by the object is equal to 0.
2: Let q be any neighbor of p with smallest layer number (modulo 4).
3: $p.down \leftarrow p$'s label for port leading to q
4: $p.layer = (q.layer + 1) \mod 4$
5: CLOCKWISE($p, p.down$)   ▷ Computes CW & CCW directions
6: **if** $p.layer$ is *odd* **then**
7:     $p.dir \leftarrow p.CW$
8: **else**
9:     $p.dir \leftarrow p.CCW$

    Extending layer $p.layer$
10: **if** the position at $p.dir$ is unoccupied, and either p is not on the first layer or p
    holds a complaint flag **then**
11:     p expands in direction $p.dir$
12:     p consumes a complaint flag, if it holds one

In the *general layering primitive*, whenever a follower is adjacent to a retired particle, it becomes a root. Root particles continue to move along positions of their layer in a clockwise (if the layer number is odd) or counter-clockwise (if the layer number is even) direction until they reach either the marked position of that layer, a retired particle in that layer, or an empty position of the previous layer (which causes them to change direction). Complaint flags are no longer needed to expand into empty positions. Followers follow their parents as before. A contracted root particle p may retire if: (i) it occupies the marked position and the marker particle in the lower layer tells it that all particles in that layer are retired (which it can determine locally), or (ii) it has a retired particle in the direction *p.dir*. Once a particle at a marked position retires, it becomes a marker particle and marks a neighboring position in the next layer as a marked position.

Algorithm 5 CLOCKWISE(p, i)

1: $j \leftarrow i$, $k \leftarrow i$
2: **while** edge j is incident to the object or to a retired particle with layer number $p.layer - 1$ **do**
3:      $j \leftarrow (j - 1) \mod 6$
4: $p.CW \leftarrow j$
5: **while** edge k is incident to the object or to a retired particle with layer number $p.layer - 1$ **do**
6:      $k \leftarrow (k + 1) \mod 6$
7: $p.CCW \leftarrow k$

Algorithm 6 MARKERRETIREDCONDITIONS(p)

First Marker Condition:
1: **if** p is the *leader* **then**
2:      p becomes a *retired* particle
3:      p sets the flag $p.marker$ to be the label of a port leading to a node guaranteed not to be in layer 1 — e.g., by taking the average direction of p's two neighbors in layer 1 (by now complete)

Extending Layer Markers:
4: **if** p is connected to a marker q and the port $q.marker$ points towards p **then**
5:      **if** both $q.CW$ and $q.CCW$ are retired **then**
6:          p becomes a *retired* particle
7:          p sets the flag $p.marker$ to the label of the port opposite the port connecting p to q

Retired Condition:
8: **if** the node in direction $p.dir$ is occupied by a retired particle **then**
9: p becomes a *retired* particle

---

3 Lower bounds

Recall that a *round* is over once every particle in P has been activated at least once. The *runtime* $T_{\mathcal{A}}(P, O)$ of a coating algorithm $\mathcal{A}$ is defined as the worst-case number of rounds (over all sequences of particle activations) required for $\mathcal{A}$ to solve the coating problem (P, O). Certainly, there are instances (P, O) where every coating algorithm has a runtime of $\Omega(n)$ (see Lemma 1), though there are also many other instances where the coating problem can be solved much faster. Since a worst-case runtime of $\Omega(n)$ is fairly large and therefore not very helpful to distinguish between different coating algorithms, we intend to study the runtime of coating algorithms relative to the best possible runtime.

Lemma 1 *The worst-case runtime required by any local-control algorithm to solve the universal coating problem is $\Omega(n)$.*

Proof Assume the particles $p_1, \ldots, p_n$ form a single line of n particles connected to the surface via p_1 (Fig. 2). Suppose $B_1 > n$. Since $d(p_n, O) = n$, it will take $\Omega(n)$ rounds in the worst-case (requiring $\Theta(n)$ movements) until p_n touches the object's surface. This worst-case can happen, for
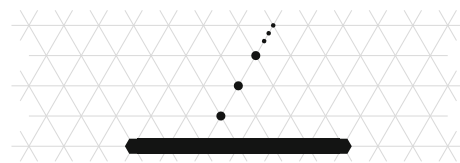
Fig. 2 Worst-case configuration concerning number of rounds. There are n particles (black dots) in a line connected to the surface via a single particle p_1

example, if p_n performs no more than one movement (either an expansion or a contraction) per round.   □

Unfortunately, a large lower bound also holds for the competitiveness of any local-control algorithm. A coating algorithm $\mathcal{A}$ is called *c-competitive* if for any valid instance (P, O),

$$E\left[T_{\mathcal{A}}(P, O)\right] \leq c \cdot \mathrm{OPT}(P, O) + K$$

where OPT(P, O) is the minimum runtime needed to solve the coating problem (P, O) and K is a value independent of (P, O).

Theorem 1 *Any local-control algorithm that solves the universal coating problem has a competitive ratio of $\Omega(n)$.*

Proof We construct an instance of the coating problem (P, O) which can be solved by an optimal algorithm in $\mathcal{O}(1)$ rounds, but requires any local-control algorithm $\Omega(n)$ times longer. Let O be a straight line of arbitrary (finite) length, and let P be a set of particles which entirely occupy layer 1, with the exception of one missing particle below O equidistant from its sides and one additional particle above O in layer 2 equidistant from its sides (see Fig. 3).

An optimal algorithm could move the particles to solve the coating problem for the given example in $\mathcal{O}(1)$ rounds, as in Fig. 4. Note that the optimal algorithm always maintains the connectivity of the particle system, so its runtime is valid even under the constraint that any connected component of particles must stay connected. However, for our local-control algorithms we allow particles to disconnect from the rest of the system.

Now consider an arbitrary local-control algorithm A for the coating problem. Given a round r, we define the
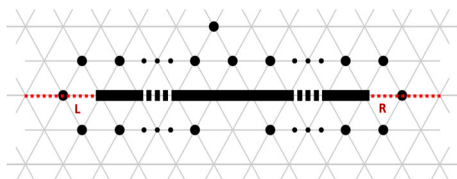
Fig. 3 The object occupies a straight line in G_{eqt}. The particles are all contracted and occupy the positions around the object, with the exception that there is one unoccupied node below the object and one extra particle above the object. Borders L and R are shown as red lines

imbalance $\phi_L(r)$ at border L as the net number of particles that have crossed L from the top of O to the bottom until round r; similarly, the imbalance $\phi_R(r)$ at border R is defined to be the net number of particles that have crossed R from the bottom of O to the top until round r.

Certainly, there is an activation sequence in which information and particles can only travel a distance of up to n / 4 nodes towards L or R within the first n / 4 rounds. Hence, for any $r \leq n/4$, the probability distributions of $\phi_L(r)$ and $\phi_R(r)$ are independent of each other. Additionally, particles up to a distance of n / 4 from L and R cannot distinguish between which border they are closer to, since the position of the gap is equidistant from the borders. This symmetry also implies that $\Pr[\phi_L(r) = k] = \Pr[\phi_R(r) = k]$ for any integer k. Let us focus on round $r = n/4$. We distinguish between the following cases.

Case 1. $\phi_L(n/4) = \phi_R(n/4)$. Then there are more particles than positions in layer 1 above O, so the coating problem cannot be solved yet.

Case 2. $\phi_L(n/4) \neq \phi_R(n/4)$. From our insights above we know that for any two values k_1 and k_2, $\Pr[\phi_L(n/4) = k_1$ and $\phi_R(n/4) = k_2] = \Pr[\phi_L(n/4) = k_2$ and $\phi_R(n/4) = k_1]$. Hence, the cumulative probability of all outcomes where $\phi_L(n/4) < \phi_R(n/4)$ is equal to the cumulative probability of all outcomes where $\phi_L(n/4) > \phi_R(n/4)$. If $\phi_L(n/4) < \phi_R(n/4)$, then there are again more particles than positions in layer 1 above O, so the coating problem cannot be solved yet.

Thus, the probability that $\mathcal{A}$ has not solved the coating problem after n / 4 rounds is at least 1 / 2, and therefore $\mathrm{E}\left[T_{\mathcal{A}}(P, O)\right] \geq 1/2 \cdot n/4 = n/8$. Since, on the other hand, OPT $= \mathcal{O}(1)$, we have established a linear competitive ratio. □

Therefore, even the competitive ratio can be very high in the worst case. We will revisit the notion of competitiveness in Sect. 5.
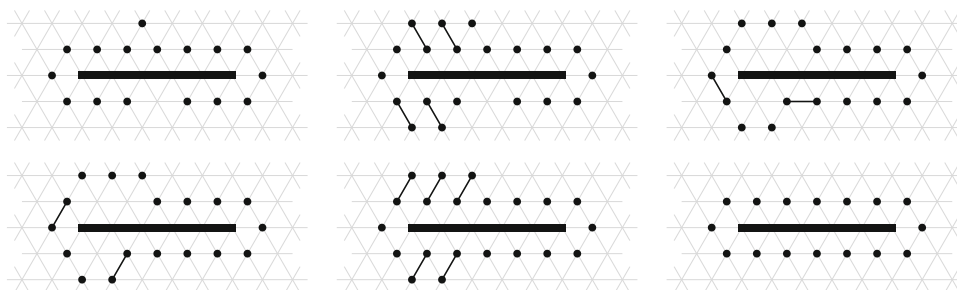
Fig. 4 Each subfigure represents the configuration of the system at the beginning of a round, and are ordered from left to right, top to bottom. After 5 rounds (i.e., at the beginning of the sixth round) the object is coated. Note that the implied algorithm can be adapted to any length of the object and always requires only 5 rounds to solve the coating problem

4 Worst-case number of rounds

In this section, we show that our algorithm solves the coating problem within a linear number of rounds w.h.p.[4]. We start with some basic notation in Sect. 4.1. Section 4.2 presents a simpler synchronous parallel model for particle activations that we can use to analyze the worst-case number of rounds. Section 4.3 presents the analysis of the number of rounds required to coat the first layer. Finally, in Sect. 4.4, we analyze the number of rounds required to complete all other coating layers, once layer 1 has been completed.

4.1 Preliminaries

We start with some notation. Recall that B_i denotes the number of nodes of G_{eqt} at distance i from object O (i.e., the number of nodes in layer i). Let N be the the layer number of the final layer for n particles (i.e., N satisfies $\sum_{j=1}^{N-1} B_j < n \leq \sum_{j=1}^{N} B_j$). Layer i is said to be *complete* if every node in layer i is occupied by a contracted retired particle (for $i < N$), or if all particles have reached their final position, are contracted, and never move again (for $i = N$).

Given a configuration C, we define a directed graph $A(C)$ over all nodes in G_{eqt} occupied by *active* (follower or root) particles in C. For every expanded active particle p in C, $A(C)$ contains a directed edge from the tail to the head of p. For every follower p, $A(C)$ has a directed edge from the head of p to $p.parent$. For the purposes of constructing $A(C)$, we also define parents for root particles: a root particle p sets $p.parent$ to be the active particle q occupying the node in direction $p.dir$ once p has performed its first handover expansion with q. For every root particle p, $A(C)$ has a directed edge from the head of p to $p.parent$, if it exists. Certainly, since every node has at most one outgoing edge in $A(C)$, the nodes of $A(C)$ form either a collection of disjoint trees or a ring of trees. A ring of trees may occur in any layer, but only temporarily; the leader election primitive ensures that a leader emerges and retires in layer 1 and marker particles emerge and retire in higher layers, causing the ring in $A(C)$ to break. The super-roots defined in Sect. 2.2 correspond to the roots of the trees in $A(C)$.

A *movement* executed by a particle p can be either a *sole contraction* in which p contracts and leaves a node unoccupied, a *sole expansion* in which p expands into an adjacent unoccupied node, a *handover contraction with p'* in which p contracts and forces its contracted neighbor p' to expand into the node it vacates, or a *handover expansion with p'* in which p expands into a node currently occupied by its expanded neighbor p', forcing p' to contract.

4.2 From asynchronous to parallel schedules

In this section, we show that instead of analyzing our algorithm for asynchronous activations of particles, it suffices to consider a much simpler model of parallel activations of particles. Define a *movement schedule* to be a sequence of particle system configurations $(C_0, \ldots, C_t)$.

Definition 1 A movement schedule $(C_0, \ldots, C_t)$ is called a *parallel schedule* if each C_i is a valid configuration of a connected particle system (i.e., each particle is either expanded or contracted, and every node of G_{eqt} is occupied by at most one particle) and for every $i \geq 0$, C_{i+1} is reached from C_i such that for every particle p one of the following properties holds:

1. p occupies the same node(s) in C_i and C_{i+1},
2. p expands into an adjacent node that was empty in C_i,
3. p contracts, leaving the node occupied by its tail empty in C_{i+1}, or
4. p is part of a handover with a neighboring particle p'.

While these properties allow at most one contraction or expansion per particle in moving from C_i to C_{i+1}, multiple particles may move in this time.

Consider an arbitrary fair asynchronous activation sequence A for a particle system, and let $C_i^{(A)}$, for $0 \leq i \leq t$, be the particle system configuration at the end of asynchronous round i in A if each particle moves according to Algorithm 1. A *forest schedule* $\mathcal{S} = (A, (C_0, \ldots, C_t))$ is a parallel schedule $(C_0, \ldots, C_t)$ with the property that $A(C_0)$ is a forest of one or more trees, and each particle p follows the unique path P_p which it would have followed according to A, starting from its position in C_0. This implies that $A(C_i)$ remains a forest of trees for every $1 \leq i \leq t$. A forest schedule is said to be *greedy* if all particles perform movements according to Definition 1 in the direction of their unique paths whenever possible.

We begin our analysis with a result that is critical to both describing configurations of particles in greedy forest schedules and quantifying the amount of progress greedy forest schedules make over time. Specifically, we show that if a forest's configuration is "well-behaved" at the start, then it remains so throughout its greedy forest schedule, guaranteeing that progress is made once every two configurations.

Lemma 2 *Given any fair asynchronous activation sequence A, consider any greedy forest schedule $(A, (C_0, \ldots, C_t))$. If every expanded parent in C_0 has at*

---

least one contracted child, then every expanded parent in C_i also has at least one contracted child, for $1 \leq i \leq t$.

Proof Suppose to the contrary that C_i is the first configuration that contains an expanded parent p which has all expanded children. We consider all possible expanded and contracted states of p and its children in C_{i-1} and show that none of them can result in p and its children all being expanded in C_i. First suppose p is expanded in C_{i-1}; then by supposition, p has a contracted child q. By Definition 1, q cannot perform any movements with its children (if they exist), so p performs a handover contraction with q, yielding p contracted in C_i, a contradiction. So suppose p is contracted in C_{i-1}. We know p will perform either a handover with its parent or a sole expansion in direction $p.dir$ since it is expanded in C_i by supposition. Thus, any child of p in C_{i-1}—say q—does not execute a movement with p in moving from C_{i-1} to C_i. Instead, if q is contracted in C_{i-1} then it remains contracted in C_i since it is only permitted to perform a handover with its unique parent p; otherwise, if q is expanded, it performs either a sole contraction if it has no children or a handover with one of its contracted children, which it must have by supposition. In either case, p has a contracted child in C_i, a contradiction.

As a final observation, two trees of the forest may "merge" when the super-root s of one tree performs a sole expansion into an unoccupied node adjacent to a particle q of another tree. However, since s is a root and thus only defines q as its parent after performing a handover expansion with it, the lemma holds in this case as well. □

For any particle p in a configuration C of a forest schedule, we define its *head distance* $d_h(p, C)$ (resp., *tail distance* $d_t(p, C)$) to be the number of edges along P_p from the head (resp., tail) of p to the end of P_p. Depending on whether p is contracted or expanded, we have $d_h(p, C) \in \{d_t(p, C), d_t(p, C) - 1\}$. For any two configurations C and C' and any particle p, we say that C *dominates* C' w.r.t. p, denoted $C(p) \succeq C'(p)$, if and only if $d_h(p, C) \leq d_h(p, C')$ and $d_t(p, C) \leq d_t(p, C')$. We say that C *dominates* C', denoted $C \succeq C'$, if and only if C dominates C' with respect to every particle. Then it holds:

Lemma 3 *Given any fair asynchronous activation sequence A which begins at an initial configuration $C_0^{(A)}$ in which every expanded parent has at least one contracted child, there is a greedy forest schedule $S = (A, (C_0, \ldots, C_t))$ with $C_0 = C_0^{(A)}$ such that $C_i^{(A)} \succeq C_i$ for all $0 \leq i \leq t$.*

Proof We first introduce some supporting notation. Let $M(p) = p^{(1)}, p^{(2)}, \ldots$ be the sequence of movements p executes according to A. Let $M_i(p)$ denote the remaining sequence of movements in $M(p)$ after the forest schedule

reaches C_i, and let $m_i(p)$ denote the first movement in $M_i(p)$.

Claim A greedy forest schedule $S = (A, (C_0, \ldots, C_t))$ can be constructed from configuration $C_0 = C_0^{(A)}$ such that, for every $0 \leq i \leq t$, configuration C_i is obtained from C_{i-1} by executing only the movements of a greedily selected, mutually compatible subset of $\{m_{i-1}(p) : p \in P\}$.

Proof Argue by induction on i, the current configuration number. C_0 is trivially obtained, as it is the initial configuration. Assume by induction that the claim holds up to C_{i-1}. W.l.o.g. let $M_{i-1} = \{m_{i-1}(p_1), \ldots, m_{i-1}(p_k)\}$, for $k \leq n$, be the greedily selected, mutually compatible subset of movements that S performs in moving from C_{i-1} to C_i. Suppose to the contrary that a movement $m'(p) \notin M_{i-1}$ is executed by a particle $p \in P$. It is easily seen that $m'(p)$ cannot be $m_{i-1}(p)$; since $m_{i-1}(p)$ was excluded when M_{i-1} was greedily selected, it must be incompatible with one or more of the selected movements and thus cannot also be executed at this time. So $m'(p) \neq m_{i-1}(p)$, and we have the cases below:

Case 1 $m_{i-1}(p)$ is a sole contraction. Then p is expanded and has no children in C_{i-1}, so we must have $m'(p) = m_{i-1}(p)$, since there are no other movements p could execute, a contradiction.

Case 2 $m_{i-1}(p)$ is a sole expansion. Then p is contracted and has no parent in C_{i-1}, so we must have $m'(p) = m_{i-1}(p)$, since there are no other movements p could execute, a contradiction.

Case 3 $m_{i-1}(p)$ is a handover contraction with q, one of its children. Then at some time in S before reaching C_{i-1}, q became a descendant of p; thus, q must also be a descendant of p in C_{i-1}. If q is not a child of p in C_{i-1}, there exists a particle $z \notin \{p, q\}$ such that q is a descendant of z, which is in turn a descendant of p. So in order for $m_{i-1}(p)$ to be a handover contraction with q, $M(z)$ must include actions which allow z to "bypass" its ancestor p, which is impossible. So q must be a child of p in C_{i-1}, and must be contracted at the time $m_{i-1}(p)$ is performed. If q is also contracted in C_{i-1}, then once again we must have $m'(p) = m_{i-1}(p)$. Otherwise, q is expanded in C_{i-1}, and must have become so before C_{i-1} was reached. But this yields a contradiction: since S is greedy, q would have contracted prior to this point by executing either a sole contraction if it has no children, or a handover contraction with a contracted child whose existence is guaranteed by Lemma 2, since every expanded parent in C_0 has a contracted child.

Case 4. $m_{i-1}(p)$ is a handover expansion with q, its unique parent. Then we must have that $m_{i-1}(q)$ is a handover contraction with p, and an argument analogous to that of Case 3 follows. □

We conclude by showing that each configuration of the greedy forest schedule $\mathcal{S}$ constructed according to the claim is dominated by its asynchronous counterpart. Argue by induction on i, the configuration number. Since $C_0 = C_0^{(A)}$, we have that $C_0^{(A)} \succeq C_0$. Assume by induction that for all rounds $0 \leq r \leq i-1$, we have $C_r^{(A)} \succeq C_r$. Consider any particle p. Since $\mathcal{S}$ is constructed using the exact set of movements p executes according to A and each time p moves it decreases either its head distance or tail distance by 1, it suffices to show that p has performed at most as many movements in $\mathcal{S}$ up to C_i as it has according to A up to $C_i^{(A)}$.

If p does not perform a movement between C_{i-1} and C_i, we trivially have $C_i^{(A)}(p) \succeq C_i(p)$. Otherwise, p performs movement $m_{i-1}(p)$ to obtain C_i from C_{i-1}. If p has already performed $m_{i-1}(p)$ according to A before reaching $C_{i-1}^{(A)}$, then clearly $C_i^{(A)}(p) \succeq C_i(p)$. Otherwise, $m_{i-1}(p)$ must be the next movement p is to perform according to A, since p has performed the same sequence of movements in the asynchronous execution as it has in $\mathcal{S}$ up to the respective rounds $i-1$, and thus has equal head and tail distances in C_{i-1} and $C_{i-1}^{(A)}$. It remains to show that p can indeed perform $m_{i-1}(p)$ between $C_{i-1}^{(A)}$ and $C_i^{(A)}$. If $m_{i-1}(p)$ is a sole expansion, then p is the super-root of its tree (in both C_{i-1} and $C_{i-1}^{(A)}$) and must also be able to expand in $C_{i-1}^{(A)}$. Similarly, if $m_{i-1}(p)$ is a sole contraction, then p has no children (in both C_{i-1} and $C_{i-1}^{(A)}$) and must be able to contract in $C_{i-1}^{(A)}$. If $m_{i-1}(p)$ is a handover expansion with its parent q, then q must be expanded in C_{i-1}. Parent q must also be expanded in $C_{i-1}^{(A)}$; otherwise $d_h(q, C_{i-1}^{(A)}) > d_h(q, C_{i-1})$, contradicting the induction hypothesis. An analogous argument holds if $m_{i-1}(p)$ is a handover contraction with one of its contracted children. Therefore, in any case we have $C_i^{(A)}(p) \succeq C_i(p)$, and since the choice of p was arbitrary, $C_i^{(A)} \succeq C_i$. □

We can show a similar dominance result when considering complaint flags.

Definition 2 A movement schedule $(C_0, \ldots, C_t)$ is called a *complaint-based parallel schedule* if each C_i is a valid configuration of a particle system in which every particle holds at most *one* complaint flag (rather than two, as described in Algorithm 3) and for every $i \geq 0$, C_{i+1} is reached from C_i such that for every particle p one of the following properties holds:

1. p does not hold a complaint flag and property 1, 3, or 4 of Definition 1 holds,
2. p holds a complaint flag f and expands into an adjacent node that was empty in C_i, consuming f,

3. p forwards a complaint flag f to a neighboring particle p' which either does not hold a complaint flag in C_i or is also forwarding its complaint flag.

A *complaint-based forest schedule* $\mathcal{S} = (A, (C_0, \ldots, C_t))$ has the same properties as a forest schedule, with the exception that $(C_0, \ldots, C_t)$ is a complaint-based parallel schedule as opposed to a parallel schedule. A complaint-based forest schedule is said to be *greedy* if all particles perform movements according to Definition 2 in the direction of their unique paths whenever possible.

We can now extend the dominance argument to hold with respect to *complaint distance* in addition to head and tail distances. For any particle p holding a complaint flag f in configuration C, we define its complaint distance $d_c(f, C)$ to be the number of edges along P_p from the node p occupies to the end of P_p. For any two configurations C and C' and any complaint flag f, we say that C *dominates* C' *w.r.t.* f, denoted $C(f) \succeq C'(f)$, if and only if $d_c(f, C) \leq d_c(f, C')$. Extending the previous notion of dominance, we say that C *dominates* C', denoted $C \succeq C'$, if and only if C dominates C' with respect to every particle and with respect to every complaint flag.

It is also possible to construct a greedy complaint-based forest schedule whose configurations are dominated by their asynchronous counterparts, as we did for greedy forest schedules in Lemma 3. Many of the details are the same, so as to avoid redundancy we highlight the differences here. The most obvious difference is the inclusion of complaint flags. Definition 2 restricts particles to holding at most one complaint flag at a time, where Algorithm 3 allows a capacity of two. This allows the asynchronous execution to not "fall behind" the parallel schedule in terms of forwarding complaint flags. Basically, Definition 2 allows a particle p holding a complaint flag f in the parallel schedule to forward f to its parent q even if q currently holds its own complaint flag, so long as q is also forwarding its flag at this time. The asynchronous execution does not have this luxury of synchronized actions, so the mechanism of buffering up to two complaint flags at a time allows it to "mimic" the pipelining of forwarding complaint flags that is possible within one round of a complaint-based parallel schedule.

Another slight difference is that a contracted particle cannot expand into an empty adjacent node unless it holds a complaint flag to consume. However, this restriction reflects Algorithm 3, so once again the greedy complaint-based forest schedule can be constructed directly from the movements taken in the asynchronous execution. Moreover, since this restriction can only cause a contracted particle to remain contracted, the conditions of Lemma 2 are still upheld. Thus, we obtain the following lemma:

Lemma 4 *Given any fair asynchronous activation sequence A which begins at an initial configuration $C_0^{(A)}$ in which every expanded parent has at least one contracted child, there is a greedy complaint-based forest schedule $\mathcal{S} = (A, (C_0, \ldots, C_t))$ with $C_0 = C_0^{(A)}$ such that $C_i^{(A)} \succeq C_i$ for all $0 \leq i \leq t$.*

By Lemmas 3 and 4, once we have an upper bound for the time it takes a greedy forest schedule to reach a final configuration, we also have an upper bound for the number of rounds required by the asynchronous execution. Hence, the remainder of our proofs will serve to upper bound the number of parallel rounds any greedy forest schedule would require to solve the coating problem for a given valid instance (P, O), where $|P| = n$. Let $\mathcal{S}^* = (A, (C_0, \ldots, C_f))$ be such a greedy forest schedule, where C_0 is the initial configuration of the particle system P (of all contracted particles) and C_f is the final coating configuration.

In Sects. 4.3 and 4.4, we will upper bound the number of parallel rounds required by $\mathcal{S}^*$ in the worst case to coat the first and higher layers, respectively. More specifically, we will bound the worst-case time it takes to complete a layer i once layers $1, \ldots, i-1$ have been completed. For convenience, we will not differentiate between complaint-based and regular forest schedules in the following sections, since the same dominance result holds whether or not complaint flags are considered. To prove these bounds, we need one last definition: a *forest–path schedule* $\mathcal{S} = (A, (C_0, \ldots, C_t), L)$ is a forest schedule $(A, (C_0, \ldots, C_t))$ with the property that all the trees of $A(C_0)$ are rooted at a path $L = v_1 v_2 \cdots v_\ell \subseteq G_{\text{eqt}}$, and each particle p must traverse L in the same direction.

4.3 First layer: complaint-based coating and leader election

Our algorithm must first organize the particles using the spanning forest primitive, whose runtime is easily bounded:

Lemma 5 *Following the spanning forest primitive, the particles form a spanning forest within $\mathcal{O}(n)$ rounds.*

Proof Initially all particles are idle. In each round any idle particle adjacent to the object, an active (follower or root) particle, or a retired particle becomes active. It then sets its parent flag if it is a follower, or becomes the root of a tree if it is adjacent to the object or a retired particle. In each round at least one particle becomes active, so—given n particles in the system—it will take $\mathcal{O}(n)$ rounds in the worst case until all particles join the spanning forest. □

For ease of presentation, we assume that the particle system is of sufficient size to fill the first layer (i.e., $B_1 \leq n$; the proofs can easily be extended to handle the case when $B_1 > n$); we also assume that the root of a tree also generates a complaint flag upon its activation (this assumption does not hurt our argument since it only increases the number of the flags generated in the system). Let $\mathcal{S}_1 = (A, (C_0, \ldots, C_{t_1}), L_1)$ be the greedy forest–path schedule where $(A, (C_0, \ldots, C_{t_1}))$ is a truncated version of $\mathcal{S}^*$, C_{t_1}—for $t_1 \leq f$—is the configuration in $\mathcal{S}^*$ in which layer 1 becomes complete, and L_1 is the path of nodes in layer 1. The following lemma shows that the algorithm makes steady progress towards completing layer 1.

Lemma 6 *Consider a round i of the greedy forest–path schedule $\mathcal{S}_1$, where $0 \leq i \leq t_1 - 2$. Then within the next two parallel rounds of $\mathcal{S}_1$, (i) at least one complaint flag is consumed, (ii) at least one more complaint flag reaches a particle in layer 1, (iii) all remaining complaint flags move one position closer to a super-root along L_1, or (iv) layer 1 is completely filled (possibly with some expanded particles).*

Proof If layer 1 is filled, (iv) is satisfied; otherwise, there exists at least one super-root in $A(C_i)$. We consider several cases:

Case 1 There exists a super-root s in $A(C_i)$ which holds a complaint flag. If s is contracted, then it can expand and consume its flag by the next round. Otherwise, consider the case when s is expanded. If it has no children, then within the next two rounds it can contract and expand again, consuming its complaint flag; otherwise, by Lemma 2, s must have a contracted child with which it can perform a handover to become contracted in C_{i+1} and then expand and consume its complaint flag by C_{i+2}. In any case, (i) is satisfied.

Case 2 No super-root in $A(C_i)$ holds a complaint flag and not all complaint flags have been moved from follower particles to particles in layer 1. Let $p_1, p_2, \ldots, p_z$ be a sequence of particles in layer 1 such that each particle holds a complaint flag, no follower child of any particle except p_z holds a complaint flag, and no particles between the next super-root s and p_1 hold complaint flags. Then, as each p_i forwards its flag to p_{i-1} according to Definition 2, the follower child of p_z holding a flag is able to forward its flag to p_z, satisfying (ii).

Case 3 No super-root in $A(C_i)$ holds a complaint flag and all remaining complaint flags are held by particles in layer 1. By Definition 2, since no preference needs to be given to flags entering layer 1, all remaining flags will move one position closer to a super-root in each round, satisfying (iii). □

We use Lemma 6 to show first that layer 1 will be filled with particles (some possibly still expanded) in $\mathcal{O}(n)$ rounds. From that point on, in another $\mathcal{O}(n)$ rounds, one

can guarantee that expanded particles in layer 1 will each contract in a handover with a follower particle, and hence all particles in layer 1 will be contracted, as we see in the following lemma:

Lemma 7 *After $\mathcal{O}(n)$ rounds, layer 1 must be filled with contracted particles.*

Proof We first prove the following claim:

Claim After $8B_1 + 2$ rounds of $\mathcal{S}$, layer 1 must be filled with particles.

Proof Suppose to the contrary that after $8B_1 + 2$ rounds, layer 1 is not completely filled with particles. Then none of these rounds could have satisfied (*iv*) of Lemma 6, so one of (*i*), (*ii*), or (*iii*) must be satisfied every two rounds. Case (*i*) can be satisfied at most B_1 times (accounting for at most $2B_1$ rounds), since a super-root expands into an unoccupied position of layer 1 each time a complaint flag is consumed. Case (*iii*) can also be satisfied at most B_1 times (accounting for at most $2B_1$ rounds), since once all remaining complaint flags are in layer 1, every flag must reach a super-root in B_1 moves. Thus, the remaining $8B_1 + 2 - 2B_1 - 2B_1 = 4B_2 + 2$ rounds must satisfy (*ii*) $2B_1 + 1$ times, implying that $2B_1 + 1$ flags reached particles in layer 1 from follower children. But each particle can hold at most one complaint flag, so at least $B_1 + 1$ flags must have been consumed and the super-roots have collectively expanded into at least $B_1 + 1$ unoccupied positions, a contradiction. □

By the claim, it will take at most $8B_1 + 2$ rounds until layer 1 is completely filled with particles (some possibly expanded). In at most another B_1 rounds, every expanded particle in layer 1 will contract in a handover with a follower particle (since $B_1 \le n$), and hence all particles in layer 1 will be contracted after $\mathcal{O}(B_1) = \mathcal{O}(n)$ rounds. □

Once layer 1 is filled, the leader election primitive can proceed. The full description of the universal coating algorithm in Derakhshandeh et al. (2017) uses a node-based version of the leader election algorithm in Daymude et al. (2015) for this primitive. For consistency, we kept this description of the primitive in this paper as well. However, with high probability guarantees were not proven for the leader election algorithm in Daymude et al. (2015). In order to formally prove with high probability results on the runtime of the universal coating algorithm, we introduced a variant of our leader election protocol under the amoebot model which provably elects a leader in $\mathcal{O}(n)$ asynchronous rounds, w.h.p.[5] (Daymude et al. 2017). This gives the following runtime bound.

---
[5] The updated leader election algorithm's runtime holds with high probability, but its correctness is guaranteed; see Daymude et al. (2017) for details.

Lemma 8 *A position of layer 1 will be elected as the leader position, and w.h.p. this will occur within $\mathcal{O}(n)$ additional rounds.*

Once a leader position has been elected and either no more followers exist (if $n \le B_1$) or all positions are completely filled by contracted particles (which can be checked in an additional $\mathcal{O}(B_1)$ rounds), the particle currently occupying the leader position becomes the leader particle. Once a leader has emerged, the particles on layer 1 retire, which takes $\mathcal{O}(B_1)$ further rounds. Together, we get:

Corollary 1 *The worst-case number of rounds for $\mathcal{S}^*$ to complete layer 1 is $\mathcal{O}(n)$, w.h.p.*

4.4 Higher layers

We again use the dominance results we proved in Sect. 4.2 to focus on parallel schedules when proving an upper bound on the worst-case number of rounds—denoted by $Layer(i)$—for building layer i once layer $i-1$ is complete, for $2 \le i \le N$. The following lemma provides a more general result which we can use for this purpose.

Lemma 9 *Consider any greedy forest–path schedule $\mathcal{S} = (A, (C_0, \ldots, C_t), L)$ with $L = v_1 v_2 \cdots v_\ell$ and any k such that $1 \le k \le \ell$. If every expanded parent in C_0 has at least one contracted child, then in at most $2(\ell + k)$ configurations, nodes $v_{\ell-k+1} \cdots v_\ell$ will be occupied by contracted particles.*

Proof Let s be the super-root closest to v_ℓ, and suppose s initially occupies node v_i in C_0. Additionally, suppose there are at least k active particles in C_0 (otherwise, we do not have sufficient particles to occupy k nodes of L). Argue by induction on k, the number of nodes in L starting with v_ℓ which must be occupied by contracted particles. First suppose that $k = 1$. By Lemma 2, every expanded parent has at least one contracted child in any configuration C_j, so s is always able to either expand forward into an unoccupied node of L if it is contracted or contract as part of a handover with one of its children if it is expanded. Thus, in at most $2(\ell + k) = 2\ell + 2$ configurations, s has moved forward ℓ positions, is contracted, and occupies its final position $v_{\ell-k+1} = v_\ell$.

Now suppose that $k > 1$ and that each node $v_{\ell-x+1}$, for $1 \le x \le k - 1$, becomes occupied by a contracted particle in at most $2(\ell + k - 1) = 2(\ell + k) - 2$ configurations. It suffices to show that $v_{\ell-k+1}$ also becomes occupied by a contracted particle in at most two additional configurations. Let p be the particle currently occupying $v_{\ell-k+1}$ (such a particle must exist since we supposed we had sufficient particles to occupy k nodes and $\mathcal{S}$ ensures the particles follow this unique path). If p is contracted in $C_{2(\ell+k)-2}$, then it remains contracted and occupying $v_{\ell-k+1}$, so we are

done. Otherwise, if p is expanded, it has a contracted child q by Lemma 2. Particles p and q thus perform a handover in which p contracts to occupy only $v_{\ell-k+1}$ at $C_{2(\ell+k)-1}$, proving the claim. □

For convenience, we introduce some additional notation. Let n_i denote the number of particles of the system that will not belong to layers 1 through $i-1$, i.e., $n_i = n - \sum_{j=1}^{i-1} B_j$, and let t_i (resp., C_{t_i}) be the round (resp., configuration) in which layer i becomes complete.

When coating some layer i, each root particle either moves either (a) through the nodes in layer i in the set direction dir (CW or CCW) for layer i, or (b) through the nodes in layer $i+1$ in the opposite direction over the already retired particles in layer i until it finds an empty position in layer i. We bound the worst-case scenario for these two movements independently in order to get a an upper bound on $Layer(i)$. Let $L_i = v_1, \ldots, v_{B_i}$ be the path of nodes in layer i listed in the order that they appear from the marker position v_1 following direction dir, and let $\mathcal{S}_i = (A, (C_{t_{i-1}+1}, \ldots, C_{t_i}), L_i)$ be the greedy forest–path schedule where $(A, (C_{t_{i-1}+1}, \ldots, C_{t_i}))$ is a section of $\mathcal{S}^*$. By Lemma 9, it would take $\mathcal{O}(B_i)$ rounds for all (a) movements to complete; an analogous argument shows that all (b) movements complete in $\mathcal{O}(B_{i+1}) = \mathcal{O}(B_i)$ rounds. This implies the following lemma:

Lemma 10 *Starting from configuration $C_{t_{i-1}+1}$, the worst-case additional number of rounds for layer i to become complete is $\mathcal{O}(B_i)$.*

Putting it all together, for layers 2 through N:

Corollary 2 *The worst-case number of rounds for $\mathcal{S}^*$ to coat layers 2 through N is $\mathcal{O}(n)$.*

Proof Starting from configuration C_{t_1+1}, it follows from Lemma 10 that the worst-case number of rounds for $\mathcal{S}^*$ to reach a legal coating of the object is upper bounded by

$$\sum_{i=2}^{N} Layer(i) \le c \sum_{i=2}^{N} B_i = \Theta(n),$$

where $c > 0$ is a constant. □

Combining Corollaries 1 and 2, we get that $\mathcal{S}^*$ requires $\mathcal{O}(n)$ rounds w.h.p. to coat any given valid object O starting from any valid initial configuration of the set of particles P. By Lemmas 3 and 4, the worst-case behavior of $\mathcal{S}^*$ is an upper bound for the runtime of our universal coating algorithm, so we conclude:

Theorem 2 *The total number of asynchronous rounds required for the universal coating algorithm to reach a legal coating configuration, starting from an arbitrary valid instance (P, O), is $\mathcal{O}(n)$ w.h.p., where n is the number of particles in the system.*

5 Simulation results

In this section we present a brief simulation-based analysis of our algorithm which shows that in practice our algorithm exhibits a better than linear average competitive ratio. Since $\text{OPT}(P, O)$ (as defined in Sect. 3) is difficult to compute in general, we investigate the competitiveness with the help of an appropriate lower bound for $\text{OPT}(P, O)$. Recall the definitions of the distances $d(p, q)$ and $d(p, U)$ for $p, q \in V_{\text{eqt}}$ and $U \subseteq V_{\text{eqt}}$. Consider any valid instance (P, O). Let $\mathcal{L}$ be the set of all legal particle positions of (P, O); that is, $\mathcal{L}$ contains all sets $U \subseteq V_{\text{eqt}}$ such that the positions in U constitute a coating of the object O by the particles in the system.

We compute a lower bound on $\text{OPT}(P, O)$ as follows. Consider any $U \in \mathcal{L}$, and let $G(P, U)$ denote the complete bipartite graph on partitions P and U. For each edge $e = (p, u) \in P \times U$, set the cost of the edge to $w(e) = d(p, u)$. Every perfect matching in $G(P, U)$ corresponds to an assignment of the particles to positions in the coating. The maximum edge weight in a matching corresponds to the maximum distance a particle has to travel in order to take its place in the coating. Let $M(P, U)$ be the set of all perfect matchings in $G(P, U)$. We define the *matching dilation* of (P, O) as

$$\text{MD}(P, O) = \min_{U \in \mathcal{L}} \left(\min_{M \in M(P,U)} \left(\max_{e \in M} (w(e)) \right) \right).$$

Since each particle has to move to some position in U for some $U \in \mathcal{L}$ to solve the coating problem, we have $\text{OPT}(P, O) \ge \text{MD}(P, O)$. The search for the matching that minimizes the maximum edge cost for a given $U \in \mathcal{L}$ can be realized efficiently by reducing it to a flow problem using edges up to a maximum cost of c and performing binary search on c to find the minimal c such that a perfect matching exists. We note that our lower bound is not tight. This is due to the fact that it only respects the distances that particles have to move but ignores the congestion that may arise, i.e., in certain instances the distances to the object might be very small, but all particles may have to traverse one "chokepoint" and thus block each other.

We implemented the universal coating algorithm in the amoebot simulator (see https://sops.engineering.asu.edu/simulations/ for videos). For simplicity, each simulation is initialized with the object O as a regular hexagon of object particles; this is reasonable since the particles need only know where their immediate neighbors in the object's border are relative to themselves, which can be determined
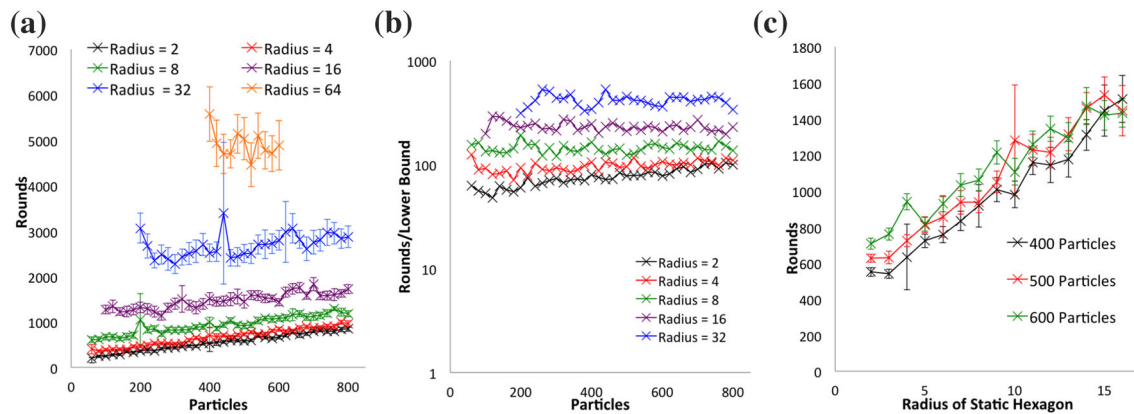
Fig. 5 **a** Shows the number of rounds varying the number of particles. **b** Shows the ratio of number of rounds to the lower bound in log scale. **c** Shows the number of rounds varying the static hexagon radius

independently of the shape of the border. The particle system P is initialized as idle particles attached randomly around the hexagon's perimeter. The parameters that were varied between instances are the radius of the hexagon and the number of (initially idle) particles in P. Each experimental trial randomly generates a new initial configuration of the system.

Figure 5a shows the number of rounds needed to complete the coating with respect to the hexagon object radius and the number of particles in the system. The number of rounds plotted are averages over 20 instances of a given $|P|$ with 95% confidence intervals. These results show that, in practice, the number of rounds required increases linearly with particle system size. This agrees with our expectations, since leader election depends only on the length of the object's surface while layering depends on the total number of particles. Figure 5b shows the ratio of the number of rounds to the matching dilation of the system. These results indicate that, in experiment, the average competitive ratio of our algorithm may exhibit closer to logarithmic behaviors. Figure 5c shows the number of rounds needed to complete the coating as the radius of the hexagon object is varied. The runtime of the algorithm appears to increase linearly with both the number of active particles and the size of the object being coated, and there is visibly increased runtime variability for systems with larger radii.

6 Conclusion

This paper continued the study of universal coating in self-organizing particle systems. The runtime analysis shows that our universal coating algorithm terminates in a linear number of rounds with high probability, and thus is worst-case optimal. This, along with the linear lower bound on the competitive gap between local and global algorithms,

further shows our algorithm to be competitively optimal. Furthermore, the simulation results indicate that the competitive ratio of our algorithm may be better than linear in practice. In the future, we would like to apply the algorithm and analysis to the case of bridging, in which particles create structures across gaps between disconnected objects. We would also like to extend the algorithm to have self-stabilization capabilities, so that it could successfully complete coating without human intervention after occasional particle failures or outside interference.

References

Angluin D, Aspnes J, Diamadi Z, Fischer MJ, Peralta R (2006) Computation in networks of passively mobile finite-state sensors. Distrib Comput 18(4):235–253

Blázovics L, Csorba K, Forstner B, Charaf H (2012a) Target tracking and surrounding with swarm robots. In: Proceedings of the 19th IEEE international conference and workshops on the engineering of computer based systems (ECBS '12), pp 135–141

Blázovics L, Lukovszki T, Forstner B (2012b) Target surrounding solution for swarm robots. In: Information and communication technologies (EUNICE '12), pp 251–262

Bonifaci V, Mehlhorn K, Varma G (2012) Physarum can compute shortest paths. J Theor Biol 309:121–133

Brambilla M, Ferrante E, Birattari M, Dorigo M (2013) Swarm robotics: a review from the swarm engineering perspective. Swarm Intell 7(1):1–41

Chen M, Xin D, Woods D (2015) Parallel computation using active self-assembly. Nat Comput 14(2):225–250

Daymude JJ, Derakhshandeh Z, Gmyr R, Strothmann T, Bazzi RA, Richa AW, Scheideler C (2015) Leader election and shape formation with self-organizing programmable matter. CoRR, abs/1503.07991, 2016. A preliminary version of this work appeared in DNA21, pp 117–132

Daymude JJ, Gmyr R, Richa AW, Scheideler C, Strothmann T (2017) Improved leader election for self-organizing programmable matter. Appeared at the 13th international symposium for algorithms and experiments for wireless networks (ALGOSEN-SORS '17). Available at CoRR:abs/1701.03616

Derakhshandeh Z, Dolev S, Gmyr R, Richa AW, Scheideler C, Strothmann T (2014) Brief announcement: amoebot—a new model for programmable matter. In: Proceedings of the 26th ACM symposium on parallelism in algorithms and architectures (SPAA '14), pp 220–222

Derakhshandeh Z, Gmyr R, Richa AW, Scheideler C, Strothmann T (2015) An algorithmic framework for shape formation problems in self-organizing particle systems. In: Proceedings of the 2nd international conference on nanoscale computing and communication (NanoCom '15), pp 21:1–21:2

Derakhshandeh Z, Gmyr R, Richa AW, Scheideler C, Strothmann T (2017) Universal coating for programmable matter. Theor Comput Sci 671:56–68

Doty D (2012) Theory of algorithmic self-assembly. Commun ACM 55(12):78–88

Kumar GP, Berman S (2014) Statistical analysis of stochastic multi-robot boundary coverage. In: Proceedings of the 2014 IEEE international conference on robotics and automation (ICRA '14), pp 74–81

Li K, Thomas K, Torres C, Rossi L, Shen C-C (2010) Slime mold inspired path formation protocol for wireless sensor networks. In: Proceedings of the 7th international conference on swarm intelligence (ANTS '10), pp 299–311

Lynch N (1996) Distributed algorithms. Morgan Kauffman, San Francisco

Michail O, Spirakis PG (2016) Simple and efficient local codes for distributed stable network construction. Distrib Comput 29(3):207–237

Patitz MJ (2014) An introduction to tile-based self-assembly and a survey of recent results. Nat Comput 13(2):195–224

Pavlic TP, Wilson S, Kumar GP, Berman S (2016) An enzyme-inspired approach to stochastic allocation of robotic swarms around boundaries. In: Inaba M, Corke P (eds) Robotics research, Springer tracts in advanced robotics, vol 114. Springer, Cham, pp 631–647

Self-organizing particle systems. https://sops.engineering.asu.edu/simulations/

Wilson S, Pavlic T, Kumar G, Buffin A, Pratt SC, Berman S (2014) Design of ant-inspired stochastic control policies for collective transport by robotic swarms. Swarm Intell 8(4):303–327

Woods D (2015) Intrinsic universality and the computational power of self-assembly. Philos Trans R Soc A Math Phys Eng Sci. https://doi.org/10.1098/rsta.2014.0214

Woods D, Chen H-L, Goodfriend S, Dabby N, Winfree E, Yin P (2013) Active self-assembly of algorithmic shapes and patterns in polylogarithmic time. In: Proceedings of the 4th conference on innovations in theoretical computer science (ITCS '13), pp 353–354