Universal Coating by 3D Hybrid Programmable Matter*

Irina Kostitsyna[1][0000−0003−0544−2257], David Liedtke[2][0000−0002−4066−0033], and Christian Scheideler[2][0000−0002−5278−528X]

[1] KBR at NASA Ames Research Center
irina.kostitsyna@nasa.gov
[2] Paderborn University, Germany
liedtke@mail.upb.de, scheideler@upb.de

Abstract. Motivated by the prospect of nano-robots that assist human physiological functions at the nanoscale, we investigate the coating problem in the three-dimensional model for hybrid programmable matter. In this model, a single agent with strictly limited viewing range and the computational capability of a deterministic finite automaton can act on passive tiles by picking up a tile, moving, and placing it at some spot. The goal of the coating problem is to fill each node of some surface graph of size n with a tile. We first solve the problem on a restricted class of graphs with a single tile type, and then use constantly many tile types to encode this graph in certain surface graphs capturing the surface of 3D objects. Our algorithm requires $\mathcal{O}(n^2)$ steps, which is worst-case optimal compared to an agent with global knowledge and no memory restrictions.

Keywords: Programmable Matter · Coating · Finite Automaton · 3D

1 Introduction

Recent advances in the field of molecular engineering gave rise to a series of computing DNA robots that are capable of performing simple tasks on the nano scale, including the transportation of cargo, communication, movement on the surface of membranes, and pathfinding [1, 3, 23, 29]. These results foreshadow future technologies in which a collective of computing particles cooperatively act as programmable matter - a homogenous material that changes its shape and physical properties in a programmable fashion. Robots may be deployed in the human body as part of a medical treatment: they may repair tissues by covering wounds with proteins or apply layers of lipids to isolate pathogens. The common thread uniting these applications is the *coating problem*, in which a thin layer of some specific substance is applied to the surface of a given object.

In the past decades, a variety of models for programmable matter has been proposed, primarily distinguished between passive and active systems. In passive

* This work was supported by the DFG Project SCHE 1592/10-1.
Due to space constraints, the proof of Lemma 5 is deferred to the full version [22].

systems, particles move and bond to each other solely by external stimuli, e.g., current or light, or by their structural properties, e.g., specific glues on the sides of the particle. Prominent examples are the DNA tile assembly models aTAM, kTAM and 2HAM (see survey in [25]). In contrast, particles in active systems solve tasks by performing computation and movement on their own. Noteworthy examples include the Amoebot model, modular self-reconfigurable robots and swarm robotics [7, 26, 31, 32]. While computing DNA robots are difficult to manufacture, passive tiles can be folded from DNA strands efficiently in large quantities [18]. A trade-off between feasibility and utility is offered by the hybrid model for programmable matter [16, 17, 19], in which a single active agent that acts as a deterministic finite automaton operates on a large set of passive particles (called tiles) serving as building blocks. A key advantage of the hybrid approach lies in the reusability of the agent upon completing a task. While agents in purely active systems are expended as they become part of the formed structure, hybrid agents can be deployed again. This property is beneficial in scenarios requiring the coating of numerous objects, such as isolating malicious cells within the human body. Coating multiple objects concurrently, with each being individually coated by a single agent, allows for efficient pipelining.

In the 3D hybrid model, we consider tiles of the shape of rhombic dodecahedra, i.e., polyhedra with 12 congruent rhombic faces, positioned at nodes of the adjacency graph of face-centered cubic (FCC) stacked spheres (see Figs. 1 and 2). In contrast to rectangular graphs (e.g., [12]), this allows the agent to fully revolve around tiles without losing connectivity, which prevents the agent and tiles from drifting apart, e.g., in liquid or low gravity environments. In this paper, we investigate the coating problem in the 3D hybrid model, in which the goal is to completely cover the surface of some impassable object with tiles, where tiles can be gathered from a material depot somewhere on the object's surface.

1.1 Our Results

We present a generalized algorithm that solves the coating problem assuming that the agent operates on a graph $G_\triangle$ of size n and degree $\Delta \leq 6$ that is a triangulation of a closed 3D surface. We assume a fixed embedding of $G_\triangle$ in $\mathbb{R}^3$ in which edges have constantly many possible orientations, and that the boundary of each node in $G_\triangle$ is a chordless cycle. Our algorithm requires only a single type of passive tiles and solves the coating problem in $\mathcal{O}(n^2)$ steps, which is worst-case optimal compared to an algorithm for an agent with global knowledge and no restriction on its memory or the number of tile types. In the 3D hybrid model, the surface graph arises as the subgraph induced by nodes adjacent to a given object (some subset of nodes) where we assume holes in the object to be sufficiently large. While that subgraph is not necessarily a triangulation with the properties described above, we show that our algorithm can be emulated on a restricted class of objects with a single type of tiles. To realize the algorithm outside of that class, we construct a virtual surface graph on which our algorithm can be emulated in $\mathcal{O}(\Delta^2 n^2)$ steps using $2^{2\Delta}$ types of passive tiles. Notably, Δ is a constant in the 3D hybrid model.

1.2 Related Work

In recent years much work on the 2D version of the hybrid model has been carried out, yet the only publication that considers the 3D variant is a workshop paper from EuroCG 2020 in which an arbitrary configuration of n tiles is rearranged into a line in $\mathcal{O}(n^3)$ steps [19]. 2D shape formation was studied in [17]; the authors provide algorithms that build an equilateral triangle in $\mathcal{O}(nD)$ steps where D is the diameter of the initial configuration. The problem of recognizing parallelograms of a specific height to length ratio was studied in [16]. The most recent publication [24] solves the problem of maintaining a line of tiles in presence of multiple agents and dynamic failures of the tiles.

Closely related to the hybrid model is the well established Amoebot model, in which computing particles move on the infinite triangular lattice via a series of expansions and contractions. In this model, a variety of problems was researched in the last years, including convex hull formation [5], shape formation [9, 11], and leader election [6]. A recent extension considers additional circuits on top of the Amoebot structure which results in a significant speedup for fundamental problems [13]. In [8,10], the authors solve the coating problem in the 2D Amoebot model; in their variant, the objective is to apply multiple layers of coating to the object. In [30], the authors solve the coating problem in the 3D Amoebot model. In their approach, the object's surface is greedily flooded by amoebots that remain connected in a tree structure. The process terminates for each amoebot when there are no more surrounding empty positions to move to. Notably, given our agent's requirement to retrieve a tile after each placement, their approach cannot be applied or transferred to the hybrid model.

Deterministic finite automata that navigate the infinite 2D grid graph $\mathbb{Z}^2$ are considered in [4, 21]. The authors of [4] address the challenge of building a compact structure, termed a nest, using available tiles. They present a solution with a time complexity of $O(s^n)$, where s denotes the initial span, and n denotes the number of tiles. In [21], the authors focus on constructing a fort, a shape with minimal span and maximum covered area, in $\mathcal{O}(n^2)$ time.

In the field of modular reconfigurable robots, coating is often part of the shape formation problem. In the 3D Catom model, a module of robots first assembles into a scaffolding [28] that is then coated by another module of robots [27]. The robots have spherical shape and reside in the FCC lattice; in contrast to the hybrid model, they assume more powerful computation, sensing and communication capabilities. The problem of leader election and local identifier assignment by generic agents in the FCC lattice is considered in [15]. Coating is approached differently in the field of swarm robotics where robots form a non-uniform spatial distribution around objects that are too heavy to be lifted alone [32].

2 Model and Problem Statement

In the *3D hybrid model*, we consider a single active agent with limited sensing and computational power that operates on a finite set of passive *tiles* positioned at nodes of some underlying graph G that has a fixed embedding in $\mathbb{R}^3$.

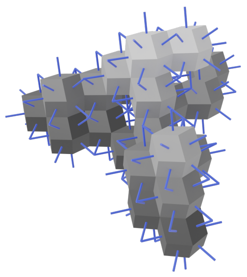
Fig. 1: Tiled nodes of the underlying graph G and their incident edges to empty nodes.

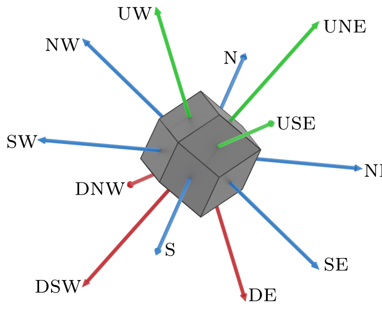

Fig. 2: A passive tile (rhombic dodecahedron) and the twelve compass directions.

2.1 Underlying Graph

Consider the close packing of equally sized spheres at each point of the infinite face-centered cubic lattice. Let $G = (V, E)$ be the adjacency graph of those spheres (see Fig. 1). This graph can be embedded in $\mathbb{R}^3$ such that nodes have alternating cubic coordinates, i.e., each node v has a coordinate $\overrightarrow{v} = (x, y, z)$ with $x, y, z \in \mathbb{Z}$ and $x + y + z$ is even. Each node has twelve neighbors whose relative positions are described by the directions UNE, UW, USE, N, NW, SW, S, SE, NE, DNW, DSW and DE, which correspond to the following vectors:

$$\begin{array}{llll} \overrightarrow{\text{UNE}} = (1,1,0), & \overrightarrow{\text{UW}} = (0,1,1), & \overrightarrow{\text{USE}} = (1,0,1), & \overrightarrow{\text{N}} = (0,1,-1), \\ \overrightarrow{\text{NW}} = (-1,1,0), & \overrightarrow{\text{SW}} = (-1,0,1), & \overrightarrow{\text{S}} = (0,-1,1), & \overrightarrow{\text{SE}} = (1,-1,0), \\ \overrightarrow{\text{NE}} = (1,0,-1), & \overrightarrow{\text{DNW}} = (-1,0,-1), & \overrightarrow{\text{DSW}} = (-1,-1,0), & \overrightarrow{\text{DE}} = (0,-1,-1). \end{array}$$

Cells in the dual graph of G w.r.t. the above embedding have the shape of rhombic dodecahedra, i.e., polyhedra with 12 congruent rhombic faces (see Fig. 2). This is also the shape of every cell in the Voronoi tessellation of G, i.e., that shape completely tessellates 3D space. Consistent to the embedding, we denote by $v + \text{X}$ the node w that is neighboring v in some direction X, i.e., $\overrightarrow{w} = \overrightarrow{v} + \overrightarrow{\text{X}}$. Consider a finite set of tiles of k distinguishable types (until Section 4 we only consider $k = 1$). Tiles have the shape of rhombic dodecahedra and are passive, in the sense that they cannot perform computation or movement on their own. A node $v \in V$ is either *tiled*, if there is a tile positioned at v, part of the object (see Section 2.3), or *empty*, otherwise. Except for the material depot (see Section 2.3), nodes can hold at most one tile at a time. We denote by $\mathcal{T}$ the set of tiled nodes, and by $\mathcal{E}$ the (infinite) set of empty nodes.

2.2 Agent Model

The agent r is the only active entity in this model. It can place and remove tiles of any type at nodes of G and loses and gains a unit of material in the process. We assume that it initially carries no material and that it can carry at most one unit of material at any time. The agent has the computational capabilities of a deterministic finite automaton performing *Look-Compute-Move* cycles. In the *look*-phase, it observes tiles at its current position p and the twelve neighbors of p, and if there are tiles, it observes their types as well. The agent is equipped with a compass that allows it to distinguish the relative positioning of its neighbors. Its initial rotation and chirality can be arbitrary, but we assume that it remains consistent throughout the execution. In the *compute*-phase the agent determines its next state transition according to the finite automaton. In the *move*-phase, the agent executes an *action* that corresponds to that state transition. It either (i) moves to an empty or tiled node adjacent to p, (ii) places a tile (of any type) at p, if $p \notin \mathcal{T}$ and r carries material (we call that *tiling* node p), (iii) removes a tile from p, if $p \in \mathcal{T}$ and r carries no material, (iv) changes the tile type at p, or (v) terminates. During (ii) and (iii), the agent loses and gains one unit of material, respectively. While the agent is technically a finite automaton, we describe algorithms from a higher level of abstraction textually and through pseudocode using a constant number of variables of constant size domain.

2.3 Problem Statement

Denote by $G(W)$ the subgraph of G induced by some set of nodes $W \subseteq V$, by $d(v, w)$ the distance (length of the shortest path) between nodes $v, w \in V$ w.r.t. G, and by $d_W(v, w)$ the distance w.r.t. $G(W)$. Consider a connected subset $\theta \subset V$ of impassable and static nodes, called *object*. Any node is either an object node, empty or tiled such that θ and the sets of empty nodes $\mathcal{E}$ and tiled nodes $\mathcal{T}$ are pairwise disjoint. A *configuration* is the tuple $C = (\mathcal{T}, \theta, p)$ containing the set of tiled nodes, object nodes, and the agent's position p. A configuration C is *valid*, if $G(\mathcal{T} \cup \theta \cup \{p\})$ is connected. We assume that holes in the object have width larger than one, i.e., $d_\theta(v, w) \leq 2$ for any $v, w \in \theta$ with $d(v, w) \leq 2$.

Let $C^0 = (\mathcal{T}^0, \theta, p^0)$ be a valid initial configuration with $\mathcal{T}^0 = \{p^0\}$. Superscripts generally refer to step numbers and may be omitted if they are clear from the context. Define the *coating layer* as the maximum subset $L \subset V \setminus \theta$ such that for each node $v \in L$ there is an object node $w \in \theta$ with $d(v, w) = 1$ and $d_L(v, p^0) < \infty$ (w.r.t. $G(L)$). The latter condition excludes unreachable nodes that are separated by the object, e.g., the inner surface of a hollow sphere. We assume a *material depot* at the agent's initial position p^0, that is p^0 is a node with the special property of holding at least $|L|$ units of material. An algorithm solves the *coating problem*, if its execution results in a sequence of valid configurations $C^0, \ldots, C^{t^*}$ such that $\mathcal{T}^{t^*} = L$, C^t results from C^{t-1} for $1 \leq t \leq t^*$ by performing any action (i)–(iv) at p^{t-1}, and the agent terminates (v) in step t^*.

3 The Coating Algorithm

In this section, we give a generalized coating algorithm for a surface graph $G_\triangle$ whose node set is the coating layer L. The agent operates only in $G_\triangle$ and all notation in this section is exclusively w.r.t. $G_\triangle$. We assume that (1) $G_\triangle$ is a triangulation of a closed 3D surface (e.g., Fig. 3a) with an embedding in $\mathbb{R}^3$ in which edges

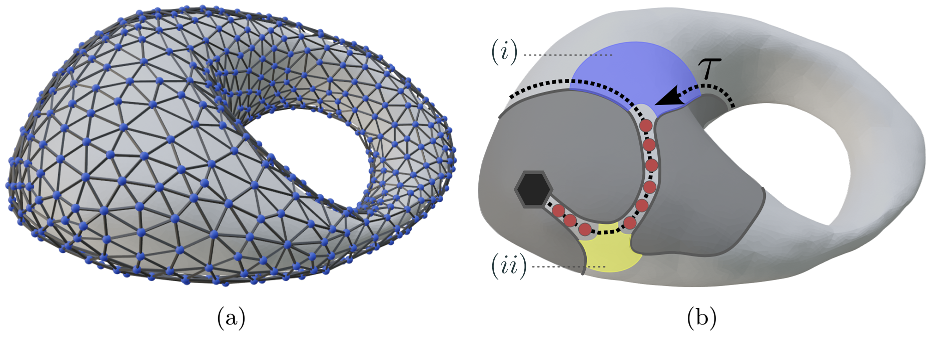

Fig. 3: (a) Example of a triangulation $G_\triangle$. (b) The simple path τ that starts at the material depot (hexagon) along the boundary of tiled nodes (opaque surface area). Links are depicted as circles. The blue area (i) is the *range* explored by the agent. In the yellow area (ii), τ is 'exposed' to nodes that are not tiled and not links. The agent explores the blue area to find an overlap with the yellow area, in which case a link can be tiled while preserving connectivity of $G_\triangle(\mathcal{E})$.

have constantly many possible orientations. Define the *i-neighborhood* $N_i(W)$ of W as the set of nodes $v \in L$ with $d(v, w) \leq i$ for some node $w \in W$, and the *boundary* as $B(W) \coloneqq N_1(W) \setminus W$. We write $N_i(w)$ and $B(w)$ for $W = \{w\}$, and use subscripts to denote subsets of only empty or only tiled nodes, e.g., $B_\mathcal{E}(w) = B(w) \cap \mathcal{E}$. Further, we assume that (2) $G_\triangle(B(v))$ contains precisely one simple cycle for any $v \in L$ (we say that $B(v)$ is *chordless*). Notably, this assumption does not weaken our results, since there are surface graphs in which properties (1) and (2) naturally arise, and otherwise we can emulate the properties using additional tile types (see Section 4).

3.1 High-Level Description and Preliminaries

The main challenge is the exploration of $G_\triangle$ to find an empty node that can be tiled while maintaining a path back to the material depot, as it is the only source of additional tiles. The exploration of all graphs of diameter D and degree Δ requires $\Omega(D \log(\Delta))$ memory bits [14]. With constant memory already the exploration of plane labyrinths (grid graphs in $\mathbb{Z}^2$) requires two pebbles (placable markers) [2,20]. Our agent has only constant memory and while it is not provided with pebbles, we can use tiles to aid the exploration of $G_\triangle$. Since at some point all nodes in $G_\triangle$ must be tiled, and our algorithm uses only a single type of tiles, i.e., tiles are indistinguishable, we cannot directly use tiles as markers. Instead, our strategy involves strategically placing tiles to create a 'narrow tunnel' of empty nodes leading to the tile depot. We then employ the left- and right-hand-rule (LHR, RHR) commonly used in labyrinth traversal to navigate through the resulting tile structure. In this process, the agent consistently maintains contact with the set of tiled nodes, either on its left or right. Essentially, we 'mark' empty nodes by disconnecting tiles in their boundary such that they become part of the tunnel. This concept is formalized through the introduction of *links*:

Definition 1. *A node $v \in \mathcal{E}$ is a* link*, if $B_{\mathcal{E}}(v)$ is disconnected. A node $v \in \mathcal{E}$* generates *(a link at) node w, if tiling node v turns w into a link. Similarly, v* consumes *(a link at) node w, if by tiling v, w is no longer a link.*

From a high level perspective, the agent traverses the boundary of tiled nodes by following the LHR until it finds a node to place its carried tile at. It then moves back to the material depot following the RHR, gathers material and repeats the process. Following the LHR and RHR, the agent cannot leave the connected component of $G_{\triangle}(\mathcal{E})$ that contains its position. Hence, it is crucial that each tile placement preserves connectivity of $G_{\triangle}(\mathcal{E})$, as otherwise the agent might never find its way back to the material depot, or it might terminate without tiling some node in L. A naive approach to maintain connectivity would be to never place a tile at a link. However, that strategy only works if the surface that is captured by $G_{\triangle}$ is simply connected, i.e., it does not contain any hole. In fact, the link property is necessary for some node $v \in \mathcal{E}$ to be a cut node w.r.t. $G_{\triangle}(\mathcal{E})$ but it is not sufficient. If the surface contains holes, then the naive approach would converge $G_{\triangle}(\mathcal{E})$ to a cyclic graph containing only links, which again may be impossible to explore. Hence, links must be tiled eventually.

In our algorithm, we build a simple path τ of empty nodes that contains all links and is completely contained in the boundary $B(\mathcal{T})$ of tiled nodes (see Fig. 3b). We cannot ensure connectivity of links on τ, i.e., there can be multiple sections at which τ is 'exposed' to empty nodes that are not links. However, we can ensure that links on τ are 'sufficiently close' to each other which we will formalize below by the notion of *segments*. In each traversal of τ, the agent visits all links precisely once by following the LHR until there are no more links in the segment at the agent's position. Afterwards, it explores a constant size neighborhood called *range*, where we define ranges such that they contain all 'close' links (all segments in some local neighborhood). The presence of a link in that range indicates that we have found one of the above mentioned 'exposures' of τ that was already visited before, i.e., there exists a link in some cycle of $G_{\triangle}(\mathcal{E})$ that can safely be tiled. Note that the agent does not necessarily traverse τ fully. As an example, the subpath of τ that follows the topmost link v in Fig. 3b is not traversed, since the next segment following node v contains no link.

Additional Terminology. The agent (at node p) maintains a direction pointer to some tiled node $a(p) \in B(p)$, called *anchor* of p. Using the analogy of wall-following in a labyrinth, $a(p)$ is the node at which the agent's 'hand' is currently placed. Initially, the agent leaves the tile depot p^0 to an arbitrary neighbor and sets its anchor to p^0. Afterwards, it can follow the LHR (RHR) by moving to the first empty node v in a clockwise (counter-clockwise) order of $B(p)$ starting at $a(p)$ and sets its anchor to the last tiled node between $a(p)$ and v in that order (see Fig. 4a). Note that the agent's anchor does not necessarily change after each move. Subsequently, L and R denote the direction to the next node according to the LHR and RHR, respectively.

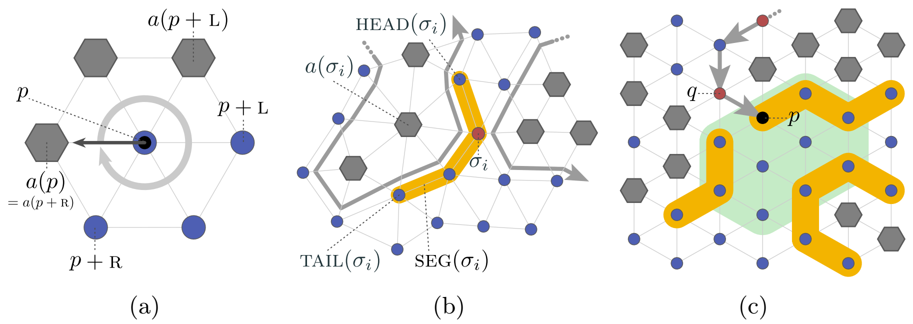

Fig. 4: Tiled nodes are depicted as hexagons, empty nodes as disks, links are depicted in red (node σ_i in (b), q in (c) and the uppermost central node in (c)). (a) Example of the updates made to the agent's anchor $a(p)$ while following the LHR and RHR. (b) $\text{SEG}(\sigma_i), \text{TAIL}(\sigma_i)$ and $\text{HEAD}(\sigma_i)$ for some node $\sigma_i \in \sigma$ in an example configuration. (c) Example of the i-range $R_i(p,q)$ for $i=2$ (orange, opaque) together with $N_2(p)$ w.r.t. $G_\triangle(\mathcal{E} \setminus \{q\})$ (green, transparent).

Let $s^0 \in B(p^0)$ be a dedicated starting node chosen arbitrarily in the initialization. Define $\sigma = (\sigma_0, \ldots, \sigma_m)$ as the path along nodes of $B(\mathcal{T})$ according to a LHR traversal that starts and ends at $s^0 = \sigma_0 = \sigma_m$. Note that σ is not necessarily a simple cycle. In fact, in a full traversal of σ, the agent can visit a link multiple times with its anchor set to different tiled nodes on each visit (e.g., σ_i in Fig. 4b). To avoid ambiguity, $a(\sigma_i)$ refers to the anchor *after* moving from σ_{i-1} to σ_i, and $a(v)$ refers to the anchor at the first occurence of v on σ.

The *segment* $\text{SEG}(\sigma_i)$ of a node σ_i (see Fig. 4b) contains all nodes that can be reached from σ_i by following the LHR or RHR while keeping the anchor fixed at $a(\sigma_i)$. Simply put, $\text{SEG}(\sigma_i)$ is the set of nodes that touch the anchor $a(\sigma_i)$ from the same 'side' as σ_i. Any node $\sigma_j \in \text{SEG}(\sigma_i)$ is a *successor* of σ_i, if $j > i$, or a *predecessor* of σ_i, if $j < i$. As an example, node σ_i in Fig. 4b has two predecessors and one successor. Denote by $\text{HEAD}(\sigma_i)$ the node without a successor in $\text{SEG}(\sigma_i)$, and $\text{TAIL}(\sigma_i)$ the node without a predecessor in $\text{SEG}(\sigma_i)$.

We can now formally introduce the above mentioned path τ. Define $\tau = (\tau_0, \ldots, \tau_l)$ as a maximal simple sub-path of σ that starts at s^0, i.e., $\tau_i = \sigma_i$ for any $0 \le i \le l$. The definition of $\text{SEG}(\cdot), \text{HEAD}(\cdot)$ and $\text{TAIL}(\cdot)$ directly carry over from σ. For simplicity, we write $v \in \tau$ or $v \in \sigma$ if τ or σ contains node v.

Finally, we introduce the aforementioned *range* that is explored by the agent before each tile placement. Consider the agent to be positioned at some empty node p, such that the node $q = p + \text{R}$ is a link. The *i-range* $R_i(p,q)$ (see Fig. 4c) is a specific neighborhood of empty nodes that is defined as if node q were tiled. Consider a node v that can be reached from p in at most i steps without moving through q or any tiled node. If the segment $\text{SEG}(v)$ does not contain q, we add it to $R_i(p,q)$. Otherwise, that segment is separated at q and only the part that contains v is added. The formal definition is as follows:

Definition 2. *The i*-range *of p w.r.t. q is the set of nodes*

$$R_i(p,q) := \bigcup_{v \in N_i(p)} \bigcup_{w \in B_{\mathcal{T}}(v)} \textsc{seg}(v)$$

where $N_i(p)$ and SEG(v) *are w.r.t. $G_\triangle(\mathcal{E} \setminus \{q\})$ and anchor w.*

3.2 Algorithm Details and Pseudocode

The coating algorithm (see pseudocode in Algorithm 1) consists of an initialization Init (lines 1–3) and two phases Coat (lines 4–10), dedicated to the tile placement, and Fetch (lines 11–16), dedicated to the gathering of material, the traversal of τ, and the termination. The agent switches between phases Coat and Fetch after each tile placement. In the pseudocode, LI and GEN denote the set of all links and generators (see Definition 1).

In phase Init (lines 1–3), the agent gathers material and moves to an arbitrary node $s^0 \in B(p^0)$. It stores the direction of p^0 w.r.t. s^0 such that it can recognize s^0 by the adjacent material depot at a later visit, and it initializes $a(s^0) \leftarrow p^0$ and SKIP $\leftarrow$ $false$. Note that s^0 is the first node of the paths σ and τ, $a(p)$ is the agent's anchor, and SKIP is a flag that indicates that the search for links in the 3-range is skipped in the next execution of phase Coat (see line 4). The flag is necessary to maintain a crucial invariant which we elaborate in the proof of Lemma 5. Afterwards, the agent moves L and enters phase Coat.

Phase Coat is always entered such that $p \notin$ LI $\cup \{s^0\}$ and $p +$ R is the last node $v \in \tau$ (i.e., with maximum index) for which $v \in$ LI$\cup\{s^0\}$. In each execution

Algorithm 1: Coating Algorithm

```
   Phase Init:
1  gather material from $p^0$; move to an arbitrary $s^0 \in B(p^0)$
2  store the direction of $p^0$ w.r.t. $s^0$; $a(s^0) \leftarrow p^0$; SKIP $\leftarrow false$
3  move L; enter phase Coat                                   ▷ $p \leftarrow s^0 + L$
   Phase Coat:
4  if SKIP or $R_3(p, p+R) \cap (LI \cup \{s^0\}) = \emptyset$ then
5      place a tile at $p$; SKIP $\leftarrow false$; enter phase Fetch
6  else
7      move R                                                 ▷ $p \leftarrow p + R$
8      if $p \in$ GEN then move R                             ▷ $p \leftarrow p + R$
9      if $p \in$ GEN and $p + R \notin LI \cup \{s^0\}$ then SKIP $\leftarrow true$
10     place a tile at $p$; enter phase Fetch
   Phase Fetch:
11 while $p \neq s^0$ do  move R                              ▷ $p \leftarrow p + R$
12 move to $p^0$; gather material from $p^0$; move to $s^0$
13 if $B_{\mathcal{E}}(s^0) = \emptyset$ then  place a tile at $s^0$; terminate
14 while $p \in LI \cup \{s^0\}$ or $v \in$ LI for a successor $v$ of $p$ in SEG$(p)$ do
15     move L                                                 ▷ $p \leftarrow p + L$
16 enter phase Coat
```

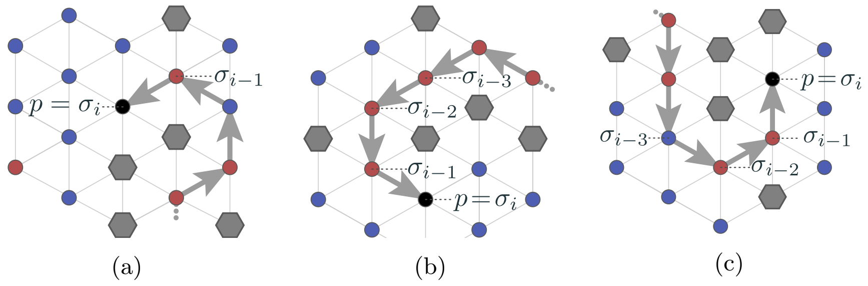

Fig. 5: Examples in which the next tile is placed at some link (red disk) in phase COAT: σ_{i-1} is tiled in (a); σ_{i-2} is tiled in (b) and (c). Only in (c), SKIP is set to *true* since $\sigma_{i-3} \notin \text{LI} \cup \{s^0\}$. As a result, σ_{i-3} is tiled on the next visit.

of phase COAT the tile that is carried by the agent is either placed directly at p (lines 4–5), or at some link that can be reached by following the RHR for at most two steps (lines 7–10). In any case, the agent enters phase FETCH afterwards. The position of the next tile depends on the following criteria: If the flag SKIP is set to *true*, then the tile is placed at p and SKIP is set to $false$ afterwards. The tile is also placed at p, if $R_3(p, p + \text{R})$ (see Definition 2) does not contain any node of $\text{LI} \cup \{s^0\}$. Otherwise, the agent must have found some link or the starting node, which implies that $R_3(p, p + \text{R})$ contains a node $v \in \tau$ with smaller index than $p \in \tau$, i.e., the agent has detected a cycle in $G_\triangle(\mathcal{E})$ in which it can safely place a tile at some link. It is crucial that no link is generated on the 'wrong side' of the newly placed tile as this link may later lead to a false detection. Let the agent's position w.r.t. σ be $p = \sigma_i$, i.e., $\sigma_{i-1}, \sigma_{i-2}$ and σ_{i-3} are the next three nodes visited by following the RHR (see Fig. 5). If $\sigma_{i-1} \notin \text{GEN}$, then node σ_{i-1} is tiled, otherwise node σ_{i-2} is tiled. In the latter case, SKIP is set to *true* whenever $\sigma_{i-2} \in \text{GEN}$ and $\sigma_{i-3} \notin \text{LI}$ (line 9). The flag SKIP ensures that node σ_{i-3} is tiled next such that the agent again enters phase COAT L of the last node $v \in \tau$ with $v \in \text{LI} \cup \{s^0\}$.

In phase FETCH, the agent moves R until it is positioned at s^0 (line 11), which it detects by the adjacent material depot and the direction stored in phase INIT. It moves to p^0, gathers material and returns to s^0 (line 12). If s^0 has no empty neighbors, it places a tile at s^0 and terminates (line 13). Otherwise, the agent moves L as long as $p \in \text{LI} \cup s^0$ or whenever a successor of p in $\text{SEG}(p)$ is a link, and switches to phase COAT afterwards (lines 14–16). In that step, the agent implicitly explores $\text{SEG}(p)$, since p can have multiple successors.

3.3 Analysis

Consider an initial configuration $C^0 = (\mathcal{T}^0, \theta, p^0)$ with $p^0 \in \mathcal{T}^0 \subseteq L$, and a material depot of size at least $|L| - |\mathcal{T}^0| + 1$ at p^0. In the problem statement we assume that p^0 is the only initially tiled node, but now we allow $\mathcal{T}^0$ to contain multiple tiled nodes besides p^0. This will later become useful in Section 4 where we construct a virtual graph in which some nodes are tiled initially. We analyze Algorithm 1 given that C^0 satisfies the following definition:

Definition 3. *A configuration $C^0 = (\mathcal{T}^0, \theta, p^0)$ is coatable w.r.t. $G_\triangle$, if $|B(v)| \leq 6$ for any $v \in \mathcal{E}^0$, $B_{\mathcal{E}^0}(p^0) \neq \emptyset$, $\textsc{li}^0 = \emptyset$, and $\mathcal{E}^0$ is connected.*

We aim to maintain five properties as invariants: ***P1***: Links may only occur on the simple path τ, i.e., $\textsc{li} \subseteq \tau$. ***P2***: All links are connected by a sequence of overlapping segments to the starting node, i.e., $\textsc{tail}(v) \in \textsc{li} \cup \{s^0\}$ for any $v \in \textsc{li}$. ***P3***: The subpath of τ from s^0 to the last link on τ induces no cycle in $G_\triangle$, i.e., for any $i < j \leq k$ with $\tau_k \in \textsc{li}$: if $d(\tau_i, \tau_j) = 1$, then $j = i + 1$. ***P4***: The boundary of any link contains precisely two connected components of empty nodes, i.e., $||B_{\mathcal{E}}(v)|| = 2$ for any $v \in \textsc{li}$ where $||\cdot||$ denotes the number of connected components. ***P5***: There exists a node of τ at which the agent enters phase Coat, i.e., either $\textsc{li} = \emptyset$ or there is an i such that $\tau_i \notin \textsc{li} \cup \{s^0\}$ and $\textsc{head}(\tau_i) = \tau_i$.

Observe that all properties hold initially by $\textsc{li}^0 = \emptyset$. The structure of our proof is as follows: We prove termination given that $P5$ is maintained, and that $\mathcal{E}$ never disconnects given that $P1$–$P4$ are maintained. Since the agent always finds a node to place the next tile at by $P5$, there must eventually be a step in which $B_{\mathcal{E}}(s^0) = \emptyset$. Since $\mathcal{E}$ remains connected until that step, $L = \mathcal{T}$ holds after the last tile is placed at s^0. Finally, we show that $P1$–$P5$ are maintained as invariants. We start with the termination of the algorithm:

Lemma 1. *If $P5$ holds in step t in which the agent gathers material at p^0, then there is a step $t^+ > t$ in which the agent enters p^0 again or terminates.*

Proof. Assume by contradiction that the agent does not terminate or enter p^0 again in any step $t' > t$. There are two cases: the agent places a tile in phase Coat and moves to a connected component of $\mathcal{E}$ that does not contain s^0, or the agent never enters phase Coat, i.e., it moves indefinitely L in phase Fetch. If the agent traverses a simple path from s^0 to some node v by moving L, places a tile at v and moves R afterwards, it must enter the connected component of $\mathcal{E}$ that contains s^0. Hence, in the first case, the agent places a tile at v after visiting v at least twice, i.e., it has fully traversed τ, which contradicts the existence of node τ_i specified by $P5$. In the second case, the agent never reaches τ_i, as it would otherwise enter phase Coat and place a tile. This implies a cycle $(\tau_j, ..., \tau_k)$ with $\tau_{k+1} = \tau_j$ and $j < i$, which contradicts that τ is a simple path. Hence, there is a step t^+ in which p^0 is entered again or $B_{\mathcal{E}}(s^0) = \emptyset$ and the agent terminates. □

Subsequently, our notation refers to some step t in which the agent gathers material at p^0. With a slight abuse of notation we will use a superscript $^+$ $(^-)$ to denote the next (previous) step t^+ (t^-) in which the agent gathers material at p^0 or terminates, e.g., $\mathcal{E}^+$ denotes the set of empty nodes in step t^+.

Lemma 2. *If $P2$ holds in step t and phase Coat is entered at some τ_i between step t and t^+, then τ_{i-1} is the last node v on τ with $v \in \textsc{li} \cup \{s^0\}$.*

Proof. As long as $p \in \textsc{li} \cup \{s^0\}$, the agent moves $\textsc{l}$ in phase Fetch which implies $\tau_{i-1} \in \textsc{li} \cup \{s^0\}$. It also moves $\textsc{l}$ if a successor of p in $\textsc{seg}(p)$ is a link. Hence, no successor v of τ_{i-1} in $\textsc{seg}(\tau_{i-1})$ is a link. Assume by contradiction that τ_{i-1} is not the last node on τ that is contained in $\textsc{li} \cup \{s^0\}$. Let τ_k be the first link after τ_{i-1}, i.e., $\tau_k \in \textsc{li}$ and $k > i-1$. Since no successor of τ_{i-1} in $\textsc{seg}(\tau_{i-1})$ is a link, τ_k cannot be contained in $\textsc{seg}(\tau_{i-1})$. Let $S = (v_0, ..., v_m)$ be the sequence of nodes with $v_0 = \tau_k$, $v_m = s^0$ and $v_j = \textsc{tail}(v_{j-1})$ for any $0 < j \leq m$. The agent's anchor changes only if there is no further successor in its current segment, i.e., *after* moving $\textsc{l}$ at some node v of τ with $\textsc{head}(v) = v$. This implies $\textsc{head}(\textsc{tail}(v)) = \textsc{tail}(v)$ for all $v \in \tau$. It follows that S must contain some node v_j with $0 < j < m$ for which $v_j = \textsc{head}(\tau_i)$. Since $\textsc{head}(\tau_i) \notin \textsc{li}$, there must exists some $v_{j'}$ with $j' < j$ for which $\textsc{tail}(v_{j'}) \notin \textsc{li} \cup \{s^0\}$ which contradicts $P2$ and concludes the lemma. □

Lemma 3. *If $\mathcal{E}$ is connected and $P1$–$P4$ hold in step t, then $\mathcal{E}^+$ is connected.*

Proof. Tiling a node $v \notin \textsc{li}$ cannot disconnect $\mathcal{E}$ by Definition 1. This covers the tile placement at the start of phase Coat, especially the case $\textsc{skip} = \textit{true}$, and the last placement before termination in phase Fetch. Thus, we must only consider cases in which a tile is placed at some link $v \in \textsc{li}$. By Lemma 2, the agent enters phase Coat at τ_i such that τ_{i-1} is the last node $w \in \tau$ with $w \in \textsc{li} \cup \{s^0\}$. Together with $P1$ follows that whenever it detects some node $w \in R_3(\tau_i, \tau_{i-1})$ with $w \in \textsc{li} \cup \{s^0\}$, then w was visited in the previous execution of phase Fetch. Let τ_j be the last node of τ that is contained in $R_3(\tau_i, \tau_{i-1})$ with $j < i-1$, and P be the shortest path from τ_i to τ_j in $G_\triangle(R_3(\tau_i, \tau_{i-1}))$. Then $C = P \circ (\tau_{j+1}, \tau_{j+2}, ..., \tau_{i-2}, \tau_{i-1})$ is a simple cycle in $G_\triangle$, where $\circ$ is the concatenation of paths. Placing a tile at τ_{i-1} or τ_{i-2} cannot disconnect C since it is a cycle, and it cannot disconnect $\mathcal{E}$ since C contains nodes of all connected components of $B_\mathcal{E}(\tau_{i-1})$ (and $B_\mathcal{E}(\tau_{i-2})$, if $\tau_{i-2} \in \textsc{li}$) by $P4$. □

Lemma 4. *If $B_\mathcal{T}(v) \cap B_\mathcal{T}(w) \neq \emptyset$, then tiling v cannot increase $||B_\mathcal{E}(w)||$.*

Proof. The lemma follows trivially if $v \notin B(w)$. Consider arbitrary $v, w \in L$ with $v \in B(w)$. Since $G_\triangle$ is a triangulation, the edge $\{v, w\}$ is contained in precisely two triangular faces, each of which contains another node u_1 and u_2, respectively. Since $B(v)$ and $B(w)$ are chordless, these are the only nodes adjacent to both v and w, which implies that $B(v) \cap B(w) = \{u_1, u_2\}$ and that $\{u_1, v, u_2\}$ is connected in $B(w)$. If any u_i is tiled, then $||B_\mathcal{T}(w) \cup \{v\}|| \leq ||B_\mathcal{T}(w)||$. Thereby, $||B_\mathcal{E}(w) \setminus \{v\}|| = ||B_\mathcal{T}(w) \cup \{v\}|| \leq ||B_\mathcal{T}(w)|| = ||B_\mathcal{E}(w)||$. □

We can now deduce the precise neighborhood of v for the case where v is both a link and a generator, i.e., $v \in \textsc{li} \cap \textsc{gen}$. By Definition 3, $B(v)$ contains at most six nodes, at least two of which must be tiled, as otherwise v cannot be a link. Hence, at most four empty nodes in at least two connected components of $B_\mathcal{E}(v)$ remain. If each connected component has size at most two, then all nodes in $B_\mathcal{E}(v)$ share a tiled neighbor with v, which contradicts that v is a generator by the previous lemma. Hence, as a corollary we obtain the following:

Corollary 1. *For any $v \in \textsc{li} \cap \textsc{gen}$: $B_{\mathcal{T}}(v)$ contains two connected components of size one, and $B_{\mathcal{E}}(v)$ contains two connected components of size one and three.*

Lemma 5. *If $P1$–$P5$ hold and a node is tiled in step t, then either $P1$–$P5$ hold in step t^+ or the agent sets $\textsc{skip} = true$ and $P1$–$P5$ hold in step t^{++}.*

The proof of Lemma 5 is deferred to the full version of the paper [22]. Essentially, we distinguish the type of node v that is tiled in step t, i.e., whether v is not a link, a link but no generator, or a link and a generator, and finally whether $v + \textsc{r} \in \textsc{li} \cup \{s^0\}$ (see line 9). In all but the last case, we can show that $P1$–$P5$ immediately hold in step t^+, and in the last case, we know by the algorithm that $\textsc{skip}$ is set to true after tiling v in step t^+. Here we can show that only property $P2$ is violated in step t^+, and only within the neighborhood of the previously placed tile. Using Corollary 1, we can precisely determine at which node the next tile is placed (with $\textsc{skip} = true$), and that this tile placement restores $P2$ in step t^{++}. In the other cases the properties mostly follow from Lemma 4.

All properties hold in an initial configuration, and they are maintained as invariants by Lemma 5. By Lemma 2, the agent eventually terminates in some step t^*, and by Lemma 3 $\mathcal{E}$ never disconnects. Hence, $\mathcal{T}^{t^*} = L$ holds after termination which concludes the following:

Theorem 1. *Following Algorithm 1, a finite-state agent solves the coating problem on $G_\triangle$, given a configuration $C^0 = (\mathcal{T}^0, \theta, p^0)$ that is coatable w.r.t. $G_\triangle$.*

3.4 Runtime Analysis

Since the agent does not sense any node outside of $N_1(p)$ in its look-phase, exploring $R_i(p,q)$ requires additional steps. From $R_i(p,q) \subset N_{i+2}(p)$ it follows that the number of steps is upper bounded by $2 \cdot |N_{i+2}(p)| = \mathcal{O}(|N(p)|^i)$. Since i is a constant and $G_\triangle$ has constant degree, each execution of phase Coat takes $\mathcal{O}(1)$ steps. Each execution of phase Fetch takes $\mathcal{O}(|\tau|)$ steps as the agent traverses a sub-path of τ twice. Since τ is simple, it follows that $|\tau| = \mathcal{O}(n)$, where $n = |L|$. The agent can place at most n tiles until $L = \mathcal{T}$, thereby performs at most n executions of Coat and Fetch, which results in $\mathcal{O}(n^2)$ steps in total.

An agent $\tilde{r}$ with unlimited memory and global vision can reach any node via a shortest path instead of sticking to the boundary of tiled nodes. Except for the last placed tile it must always return to the material depot which implies that the last tile is placed at a node w with maximum distance to p^0. In the worst case, the surface graph is the triangulation of an object resembling a straight line such that $d_L(p^0, w) = \Theta(n)$. Each node on the shortest path P from p^0 to w must be tiled. Hence, $\tilde{r}$ takes at least $2\left(\sum_{u \in P} d_L(p^0, u)\right) - d_L(p^0, w) = \left(\sum_{i=1}^{\Theta(n)} 2 \cdot i\right) - \Theta(n) = \Theta(n^2)$ steps which implies worst-case optimality of Algorithm 1.

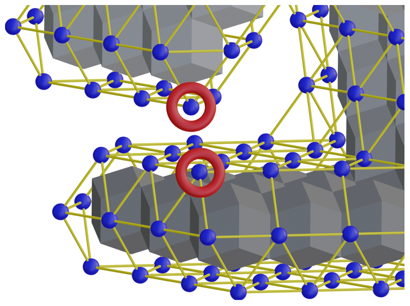
Fig. 6: A snapshot of $G_\Diamond$: the circled nodes are adjacent in $G(L)$.

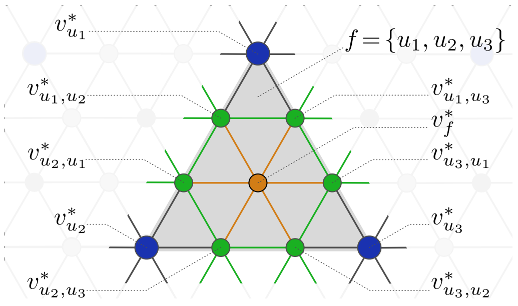

Fig. 7: A triangular face f of $G_\triangle$ and its corresponding virtual edges and nodes in $G^*_\triangle$.

4 Coating in the 3D Hybrid Model

In this section, we apply our coating algorithm to the 3D hybrid model. We first define a triangulation on nodes of L with degree $\Delta \leq 8$, and afterwards construct a virtual graph on which we emulate our algorithm using $2^{2\Delta}$ types of tiles.

Recall the definition of graph $G = (V, E)$ and its embedding in $\mathbb{R}^3$ from Section 2. Define $G_\Diamond = (L, E')$ as the subgraph of $G(L)$ that contains only those edges $\{v, w\}$ for which v and w share adjacent object neighbors (see Fig. 6), i.e., $E' = \{\{v, w\} \mid d_\theta(N_1(v), N_1(w)) \leq 1\}$. We can view $G_\Diamond$ as embedded on the surface of our 3D object. That embedding contains triangular and tetragonal faces (see Fig. 6) where tetragonal faces can occur in one of three orientations: (1) $v, v+\text{NE}, v+\text{NE}+\text{USE}, v+\text{USE}$, (2) $v, v+\text{NW}, v+\text{NW}+\text{UNE}, v+\text{UNE}$, and (3) $v, v+\text{N}, v+\text{N}+\text{UW}, v+\text{UW}$. Apart from rotation, Fig. 8 shows all possible arrangements of faces within $G_\Diamond$. We define the class of *smooth objects* $\mathcal{S}$ as all objects for which $G_\Diamond$ contains only the cases (a)–(f) from Fig. 8. Let $G_\triangle$ be the triangulation of $G_\Diamond$ in which the same diagonal edge is added for each tetragonal face of the same orientation (1)–(3) (since we want the agent to be able to deduce

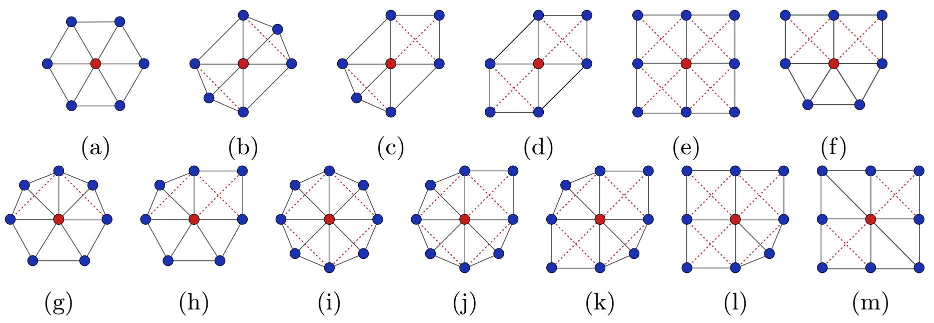

Fig. 8: All possible arrangements of faces in $G_\Diamond$ apart from rotation. Dashed edges indicate that the distance between its endpoints is precisely two w.r.t. G.

the triangulation). Since $d_L(v,w) = 1$ w.r.t. $G_\triangle$ implies $d_L(v,w) \leq 2$ w.r.t. $G_\Diamond$, the agent can emulate moving on $G_\triangle$ with a multiplicative time and memory overhead of at most two. It is easy to see that $B(v)$ is chordless and v has degree at most six for all $v \in L$ within the class $\mathcal{S}$. Together with Theorem 1 follows:

Theorem 2. *A finite-state agent with a single tile type solves the coating problem on any object $\theta \in \mathcal{S}$ with coating layer L in $\mathcal{O}(n^2)$ steps, where $n = |L|$.*

4.1 Emulation of Coatable Surface Graphs

Consider an arbitrary triangulation $G_\triangle = (L, E)$ of constant degree Δ and an initially valid configuration C_0. We construct a virtual graph $G^*_\triangle = (L^*, E^*)$ with virtual initial configuration C^{0*} such that $G^*_\triangle$ is coatable w.r.t. C^{0*}. During that construction, we define a partial surjective function $\mathcal{R} : L^* \to L$ that maps virtual nodes to real nodes. We show that an agent r operating on $G_\triangle$ w.r.t. C_0 with $2^{2\Delta}$ tile types can emulate an agent r^* that executes Algorithm 1 on $G^*_\triangle$ w.r.t. C^{0*} such that throughout the emulation $\mathcal{R}(p^*) = p$.

Virtual Graph Construction The virtual graph $G^*_\triangle$ is the result of subdividing each face of $G_\triangle$ into nine triangular faces (see Fig. 7). The node set L^* contains a virtual node v^*_u for each node $u \in L$, two virtual nodes $v^*_{u,w}$ and $v^*_{w,u}$ for each edge $\{u,w\} \in E$, and a virtual node v^*_f for each triangular face f of $G_\triangle$. For each edge $\{u,w\} \in E$ the edge set E^* contains three virtual edges $\{v^*_u, v^*_{u,w}\}$, $\{v^*_{u,w}, v^*_{w,u}\}$ and $\{v^*_{w,u}, v^*_w\}$. For each triangular face $f = \{u_1, u_2, u_3\}$ of $G_\triangle$, E^* contains six virtual edges $\{v^*_f, v^*_{u_i,u_j}\}$ and three virtual edges $\{v^*_{u_i,u_j}, v^*_{u_i,u_k}\}$, where $u_i, u_j, u_k \in f$ are pairwise distinct. We define $\mathcal{R}(v^*_{u,w}) = u$ for any virtual node $v^*_{u,w} \in L^*$. Consider an arbitrary but fixed order on the vectors $\overrightarrow{\mathrm{x}_1}, ..., \overrightarrow{\mathrm{x}_m}$ that correspond to edges in the embedding of $G_\triangle$. Let π represent that order, i.e., $\pi(\overrightarrow{\mathrm{x}_i}) = i$. For some face $f = \{u_1, u_2, u_3\}$ of $G_\triangle$, we define $\mathcal{R}(v^*_f) = u_i$, where u_i is the node minimizing $\pi(\overrightarrow{u_i} - \overrightarrow{u_j})$ for any $u_i, u_j \in f$ with $i \neq j$. We define the virtual initial configuration C^{0*} such that all v^*_u are tiled, i.e., $\mathcal{T}^{0*} = \cup_{u \in L} v^*_u$, $p^{0*} = v^*_{p^0}$ and assume a material depot of size at least $|L^*| - |L|$ at $v^*_{p^0}$.

Lemma 6. *C^{0*} is coatable w.r.t. $G^*_\triangle$.*

Proof. Each face of $G_\triangle$ is triangular, and two virtual nodes are added for each edge of $G_\triangle$. Hence, $|B(v^*_f)| = 6$ w.r.t. $G^*_\triangle$ for any face f of $G_\triangle$. Any $v^*_{u,w}$ is adjacent to $v^*_{f_1}$ and $v^*_{f_2}$, where f_1, f_2 are the two faces of $G_\triangle$ that both contain u and w, to two nodes v^*_{u,w_1}, v^*_{u,w_2}, where $w_1 \in f_1$ and $w_2 \in f_2$, and to $v^*_{w,u}$ and v^*_u. Hence, $|B(v^*_{u,w})| = 6$ w.r.t. $G^*_\triangle$ for any edge $\{u,w\}$ of $G_\triangle$. Any other virtual node is initially tiled, which implies $|B(v^*)| \leq 6$ for any $v^* \in \mathcal{E}^*$. By construction, each initially tiled node is isolated, i.e., $d(v^*, w^*) \geq 3$ for any $v^*, w^* \in \mathcal{T}^{0*}$. Since $G^*_\triangle$ is connected, it follows that $\mathcal{E}^{0*}$ is connected and $B_\mathcal{E}(v^*)$ is connected for any $v^* \in \mathcal{E}^{0*}$, i.e., $\textsc{li}^{0*} = \emptyset$. Hence, each property of Definition 3 is satisfied. □

Lemma 7. *A finite-state agent can emulate Algorithm 1 on $G^*_\triangle$ in $\mathcal{O}(\Delta^2 n^2)$ steps while moving and placing tiles of at most $2^{2\Delta}$ types on $G_\triangle$.*

Proof. Let $F^* \subset L^*$ be the set of virtual nodes v^*_f that correspond to some face f of $G_\triangle$ in the construction of $G^*_\triangle$. Since $G^*_\triangle(L^* \setminus F^*)$ is a subdivision of $G_\triangle$, it can be embedded in the same 3D surface as $G_\triangle$ using vectors that are collinear to vectors in the embedding of $G_\triangle$. It follows that we can use the same fixed order π from the construction of $G^*_\triangle$.

In the following, we define for each node $u \in L$ a bit-sequence $x(u) = (x_1, ..., x_{2\Delta})$ that encodes the occupation of all nodes $v^* \in L^*$ with $\mathcal{R}(v^*) = u$, where a 0 encodes an empty, and a 1 encodes an occupied virtual node. By the construction of $G^*_\triangle$, there are at most 2Δ nodes v^* with $\mathcal{R}(v^*) = u$ such that 2Δ bits suffice. The order of bits in $x(u)$ is uniquely given by π where the first Δ bits encode virtual nodes that correspond to edges of $G_\triangle$, and the following bits encode virtual nodes that correspond to faces of $G_\triangle$. There is no bit for the virtual node $v^*_u \in L^*$ since it is initially occupied and remains occupied until termination by following Algorithm 1. In fact, $\mathcal{R}$ is undefined for $v^*_u \in L^*$.

Consider an agent r on $G_\triangle$ that utilizes $k = 2^{2\Delta}$ types of passive tiles. Each tile type uniquely describes a bit-sequence of length $\log(k) = 2\Delta$ such that r emulates an agent r^* on $G^*_\triangle$ with initial configuration C^{0*} as follows: If r^* moves from v^* to w^*, then r moves from $\mathcal{R}(v^*)$ to $\mathcal{R}(w^*)$ (if $\mathcal{R}(v^*) \neq \mathcal{R}(w^*)$). If r^* places a tile at v^* and $\mathcal{R}(v^*)$ is empty, then r places a tile at $\mathcal{R}(v^*)$ that corresponds to the bit-sequence x in which only v^* is encoded as occupied, otherwise r incorporates the occupation of v^* by changing the tile type. If r^* gathers material and r carries no material, then r also gathers material.

By Theorem 1 and Lemma 6, r^* solves the coating problem on $G^*_\triangle$. Since $\mathcal{R}$ is surjective and any node $\mathcal{R}(v^*) \in L$ is occupied, if $v^* \in L^*$ is occupied, the emulation solves the coating problem on $G_\triangle$ in $\mathcal{O}(|L^*|^2) = \mathcal{O}(\Delta n)$ steps. □

Our final theorem follows from the virtual graph construction on top of our triangulation $G_\triangle$ of $G_\Diamond$ (with $\Delta \leq 8$) and the previous lemma:

Theorem 3. *A finite-state agent utilizing constantly many tile types can solve the coating problem on arbitrary objects in worst-case optimal $\mathcal{O}(n^2)$ steps.*

5 Future Work

We provided an algorithm that solves the coating problem in the 3D hybrid model in worst-case optimal $\mathcal{O}(n^2)$ steps given that the initial configuration w.r.t. the surface graph fulfills the property of coatability as specified in Definition 3. While the algorithm solves the problem directly in the class of smooth objects, there are certainly surface graphs that violate coatability. We bypassed this problem by emulating our algorithm on a subdivision of these surface graphs using $2^{2\Delta}$ types of tiles. A natural question for future work is whether solving the problem with a single tile type is in fact impossible, and if so, then what is the lowest number of tile types required to solve it. Another open question is how far our worst-case optimal solution is off from the best case solution.

References

1. Akter, M., Keya, J.J., Kayano, K., Kabir, A.M.R., Inoue, D., Hess, H., Sada, K., Kuzuya, A., Asanuma, H., Kakugo, A.: Cooperative cargo transportation by a swarm of molecular machines. Science Robotics **7**(65), eabm0677 (2022)
2. Blum, M., Kozen, D.: On the power of the compass (or, why mazes are easier to search than graphs). In: 19th Annual Symposium on Foundations of Computer Science (sfcs 1978). pp. 132–142 (1978). https://doi.org/10.1109/SFCS.1978.30
3. Chao, J., Wang, J., Wang, F., Ouyang, X., Kopperger, E., Liu, H., Li, Q., Shi, J., hu, J., Wang, L., Huang, W., Simmel, F., Fan, C.: Solving mazes with single-molecule dna navigators. Nature Materials **18** (2019). https://doi.org/10.1038/s41563-018-0205-3
4. Czyzowicz, J., Dereniowski, D., Pelc, A.: Building a nest by an automaton. Algorithmica **83** (2021). https://doi.org/10.1007/s00453-020-00752-0
5. Daymude, J.J., Gmyr, R., Hinnenthal, K., Kostitsyna, I., Scheideler, C., Richa, A.W.: Convex hull formation for programmable matter. In: Proceedings of the 21st International Conference on Distributed Computing and Networking. ICDCN 2020 (2020). https://doi.org/10.1145/3369740.3372916
6. Daymude, J., Gmyr, R., Richa, A., Scheideler, C., Strothmann, T.: Improved leader election for self-organizing programmable matter. In: Algorithms for Sensor Systems - 13th International Symposium on Algorithms and Experiments for Wireless Sensor Networks, ALGOSENSORS 2017, Revised Selected Papers. pp. 127–140 (2017). https://doi.org/10.1007/978-3-319-72751-6_10
7. Derakhshandeh, Z., Dolev, S., Gmyr, R., Richa, A.W., Scheideler, C., Strothmann, T.: Amoebot - a new model for programmable matter. In: Proceedings of the 26th ACM Symposium on Parallelism in Algorithms and Architectures. p. 220–222. SPAA '14 (2014). https://doi.org/10.1145/2612669.2612712
8. Derakhshandeh, Z., Gmyr, R., Porter, A.M., Richa, A.W., Scheideler, C., Strothmann, T.: On the runtime of universal coating for programmable matter. Natural Computing **17**, 81–96 (2016). https://doi.org/10.1007/s11047-017-9658-6
9. Derakhshandeh, Z., Gmyr, R., Richa, A.W., Scheideler, C., Strothmann, T.: Universal shape formation for programmable matter. In: Proceedings of the 28th ACM Symposium on Parallelism in Algorithms and Architectures. p. 289–299. SPAA '16 (2016). https://doi.org/10.1145/2935764.2935784
10. Derakhshandeh, Z., Gmyr, R., Richa, A.W., Scheideler, C., Strothmann, T.: Universal coating for programmable matter. Theoretical Computer Science **671**, 56–68 (2017). https://doi.org/https://doi.org/10.1016/j.tcs.2016.02.039
11. Di Luna, G.A., Flocchini, P., Santoro, N., Viglietta, G., Yamauchi, Y.: Shape formation by programmable particles. Distributed Computing **33**, 69–101 (2019)
12. Fekete, S., Gmyr, R., Hugo, S., Keldenich, P., Scheffer, C., Schmidt, A.: Cadbots: Algorithmic aspects of manipulating programmable matter with finite automata. Algorithmica **83**, 1–26 (2021). https://doi.org/10.1007/s00453-020-00761-z
13. Feldmann, M., Padalkin, A., Scheideler, C., Dolev, S.: Coordinating amoebots via reconfigurable circuits. Journal of Computational Biology **29** (2022). https://doi.org/10.1089/cmb.2021.0363
14. Fraigniaud, P., Ilcinkas, D., Peer, G., Pelc, A., Peleg, D.: Graph exploration by a finite automaton. In: Mathematical Foundations of Computer Science 2004. pp. 451–462 (2004)
15. Gastineau, N., Abdou, W., Mbarek, N., Togni, O.: Leader election and local identifiers for 3d programmable matter. Concurrency and Computation: Practice and Experience **34** (2020). https://doi.org/10.1002/cpe.6067

16. Gmyr, R., Hinnenthal, K., Kostitsyna, I., Kuhn, F., Rudolph, D., Scheideler, C.: Shape recognition by a finite automaton robot. In: 43rd International Symposium on Mathematical Foundations of Computer Science (MFCS 2018). LIPIcs, vol. 117, pp. 52:1–52:15 (2018). https://doi.org/10.4230/LIPIcs.MFCS.2018.52
17. Gmyr, R., Hinnenthal, K., Kostitsyna, I., Kuhn, F., Rudolph, D., Scheideler, C., Strothmann, T.: Forming tile shapes with simple robots. Natural Computing **19** (2020). https://doi.org/10.1007/s11047-019-09774-2
18. Heuer-Jungemann, A., Liedl, T.: From dna tiles to functional dna materials. Trends in Chemistry **1**(9), 799–814 (2019). https://doi.org/https://doi.org/10.1016/j.trechm.2019.07.006
19. Hinnenthal, K., Rudolph, D., Scheideler, C.: Shape formation in a three-dimensional model for hybrid programmable matter. In: Proc. of the 36th European Workshop on Computational Geometry (EuroCG 2020) (2020)
20. Hoffmann, F.: One pebble does not suffice to search plane labyrinths. In: International Symposium on Fundamentals of Computation Theory (1981)
21. Kant, K., Pattanayak, D., Mandal, P.S.: Fort formation by an automaton. In: 2021 International Conference on COMmunication Systems and NETworkS. pp. 540–547 (2021). https://doi.org/10.1109/COMSNETS51098.2021.9352839
22. Kostitsyna, I., Liedtke, D., Scheideler, C.: Universal coating by 3d hybrid programmable matter (2024). https://doi.org/arXiv.2303.16180
23. Li, H., Gao, J., Cao, L., Xie, X., Fan, J., Wang, H., Wang, H., Nie, Z.: A dna molecular robot autonomously walking on the cell membrane to drive the cell motility. Angewandte Chemie International Edition **60** (2021). https://doi.org/10.1002/anie.202108210
24. Nokhanji, N., Flocchini, P., Santoro, N.: Dynamic line maintenance by hybrid programmable matter. International Journal of Networking and Computing **13**(1), 18–47 (2023). https://doi.org/10.15803/ijnc.13.1_18
25. Patitz, M.: An introduction to tile-based self-assembly and a survey of recent results. Natural Computing **13** (2013). https://doi.org/10.1007/s11047-013-9379-4
26. Tan, N., Hayat, A.A., Elara, M.R., Wood, K.L.: A framework for taxonomy and evaluation of self-reconfigurable robotic systems. IEEE Access **8**, 13969–13986 (2020). https://doi.org/10.1109/ACCESS.2020.2965327
27. Thalamy, P., Piranda, B., Bourgeois, J.: 3d coating self-assembly for modular robotic scaffolds. In: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 11688–11695 (2020). https://doi.org/10.1109/IROS45743.2020.9341324
28. Thalamy, P., Piranda, B., Lassabe, F., Bourgeois, J.: Scaffold-based asynchronous distributed self-reconfiguration by continuous module flow. In: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 4840–4846 (2019). https://doi.org/10.1109/IROS40897.2019.8967775
29. Thubagere, A.J., Li, W., Johnson, R.F., Chen, Z., Doroudi, S., Lee, Y.L., Izatt, G., Wittman, S., Srinivas, N., Woods, D., Winfree, E., Qian, L.: A cargo-sorting dna robot. Science **357**(6356), eaan6558 (2017)
30. Traversat, W.: Universal Coating by Programmable Matter in 3D. Master's thesis, Eindhoven University of Technology (2020), https://pure.tue.nl/ws/portalfiles/portal/168210057/Traversat_W..pdf
31. Tucci, T., Piranda, B., Bourgeois, J.: A distributed self-assembly planning algorithm for modular robots. In: Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems. p. 550–558. AAMAS '18 (2018)
32. Werfel, J., Petersen, K., Nagpal, R.: Designing collective behavior in a termite-inspired robot construction team. Science **343**(6172), 754–758 (2014)