Honda Research Institute EU
Honda Research Institute Europe GmbH
https://www.honda-ri.de/

A Leader Based Coating Algorithm for Simple and Cave Shaped Objects with Robot Swarms

Ahmad Reza Cheraghi, Kalman György Graffi

2020

Preprint:

This is an accepted article published in 5th Asia-Pacific Conference on Intelligent Robot Systems (ACIRS 2020). The final authenticated version is available online at: https://doi.org/10.1109/ACIRS49895.2020.9162610
Copyright 2020 IEEE

A Leader Based Coating Algorithm for Simple and Cave Shaped Objects with Robot Swarms

Ahmad Reza Cheraghi
Technology of Social Networks
Heinrich Heine University
Düsseldorf, Germany
ahmad.cheraghi@hhu.de
https://tsn.hhu.de

Kalman Graffi
Honda Research Institute Europe GmbH
Offenbach am Main, Germany
kalman.graffi@honda-ri.de
https://www.honda-ri.de

***Abstract*—Self-organizing swarm systems promise an essential progress in several research areas. They can find application in health context, such as through nano-robots, or material context, such as for health-healing properties. Swarm behavior in nature is well known e.g. bird flocks that protect individual birds, ants that collaboratively transport bigger objects, or predators who hunt in packs. Applying these strategies to future robot swarms promises to open further opportunities. In this paper, we address the problem of object coating, in which a swarm of "subjects" aims to position itself as tightly as possible around an object. We introduce a leader-based coating algorithm that promises near-optimal performance. A leader is elected to coordinate the actions of the swarm; thus, it scans the entire object and plans the placement of the individual swarm members upfront. Through its coordination, each individual subject is placed around the scanned object. While simple shapes are easy to master, a strong emphasis of this paper lies on dealing with cave-shaped objects in which passages or trails might be blocked in a naive approach. We elaborate how to deal with such problems and how our algorithm addresses this issue. In the evaluation we showcase that our approach is able to coat cave shaped objects with a 100% success rate.**

***Keywords*—Robot, Swarms, Coating, Caging, Boundary, Coverage, Scanning**

I. Introduction

Cancer is still a deadly disease that is hard to cure. For the future it is envisioned to cure this disease by using a swarm of nanobots that will discover, identify and surround the cancer cell to destroy it and stop its further growth. Such a perfect surrounding of an object is termed "coating". A swarm is the collective behavior of many entities, such as animals, robots, or even human being, for handling actions together that either cannot be done through individuals or in which the multitude of participants highly adds to the efficiency through collaboration gains. For an optimal coating, obviously, a multitude of participants are needed that interact with each other, decide on their movements and thus collaboratively engage to cover the object, here the cancer cell, with as many layers of nanobots as possible. Thus, the swarm needs to coordinate its collective behavior. This can be either done centrally, decentralized, or by having a leader. By the central way, a system outside the of the swarm is controlling the individuals in the swarm. This can be for example the player in a chest game. Generally, someone or something that is looking from above or outside of the swarm. In the decentralized way all the subjects in the swarm are equal and the individuals need to interact with each other to decide on their individual movements hopefully in a way that leads to the emergence of desired swarm behavior. An example for this approach are fish that are use their lateral senses to sense proximity and move together as close packed flock in the water to save them from predators. Finally, in a leader-based swarm, one of the subjects in the swarm is a leader and coordinates the actions of the swarm. The leader will be either elected within the swarm or based on its higher capabilities compared to the other ones such as in dominance hierarchies. Leader subjects having higher capabilities compared to others are called sometimes active elements. An example is a mother animal that carries its babies around. Its ability is to move on its own and control its action. The leader (active subject) will coordinate the actions and is responsible for the rest of the subjects. In contrast, the passive subject cannot not anything by their own and must be led.

The contribution of this paper is a leader-based coating algorithm, that, given an object formation and a swarm of subjects, will coat that object with the individuals of the swarm as tightly as possible, i.e. in a way that each individual has at maximum a distance to the object as the closest free spot around the object. In this model, we assume that the swarm of subjects, such as robots, cooperate and are capable to move and to sense their local proximity. The object formation is a two-dimensional shape, e.g. a disc, triangle, etc, that will be coated by this swarm with the help of its leader. Beside a simple shaped object, the focus of the paper is to coat cave shaped objects. A cave shaped object has a hole in its border that leads inside the cave. These types of objects bring some problems, such as it is essential to not block the entry to the cave too early. In the following, we discuss related work, elaborate the problem statement and terminology, present our leader-based approach and then prove its validation.

II. Problem Statement

The **Coating Problem** is the positioning of a given amount of *subjects* around a shaped *object* so that all the subjects are as close as possible to the object. A coating is valid, when all closest locations around the object are occupied by the subjects and the subjects are as near as possible to the object.

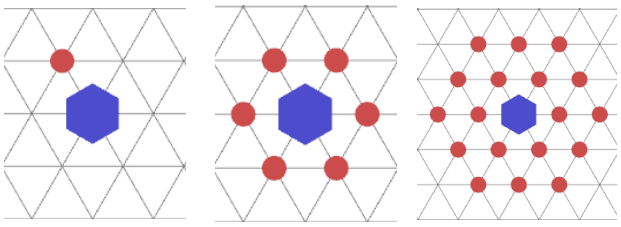
To abstract the problem space, we assume that the world is discretized in fixed locations and a location pattern, such as a hexagonal grid, is in place. Each location has then six neighboring locations. When an object occupies space it naturally exposes neighboring (discrete) locations around itself, that, depending on the number of available subjects are more or less tightly packed. Fig. 1 shows the different types of possible valid coating depending on the number of available subjects. The red dots are representing the subjects and the blue tiles the object. They are three types of (valid) coating states. The first state is the legal state (Fig. 1a), this is when the object is coated, but there are not enough subjects to coat the entire object. Next, is the Ideal state (Fig. 1b). Here, the amount of Subjects are exactly sufficient to coat the entire Object with one layer. The last state is the Layered state (Fig. 1c). This state occurs when the amount of the Subjects is higher than the number of direct locations around the object. Therefore, new layers of coating occur around the Subjects that already coated the Object.

While convex object shapes, i.e. filled, are quite easy to coat, as each location to occupy by an subject can be directly accessed, we focus also on coating cave-shaped objects. The difference from a simple object such as those in Fig. 1 is that the cave-shaped object has an entrance at its borderlines. An example is depicted in Fig. 2a. Here the green subject is in front of a cave entrance, which is the only entry to the cave. The subject must not stop there, although it is a valid coating position, in order to not block the path inside the cave. The diameter d_e of this entrance is at least as big as the diameter of a subject d_s but should not exceed the diameter of two subjects. Certainly, an entrance should fulfill this criteria $d_s \leq d_e < 2 * d_s$ criteria as shown in Fig. 2b.

Please note that shapes with entry sizes larger than two subjects are coated layer by layer until only an entry size of one remains, so that our dedicated algorithm for this use case can be applied.

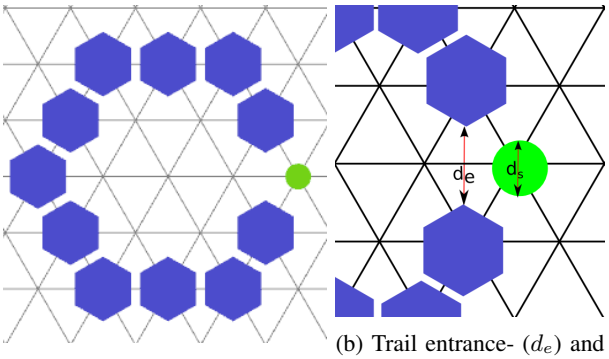
As result, this object type will bring four types of problems which we define

Challenge 1. *Recognizing a trail entrance.* If a subject is incapable of recognizing an entrance when scanning, it might coat the entrance, block thus the entry and an invalid coating occurs. This type of invalid coating is shown in Fig. 3a. The blue subjects have coated the first layer and red subjects, the second layer. The subjects marked in green green marked is the source of the problem as it has blocked the cave entrance. Consequently the inner part of the cave cannot be coated.



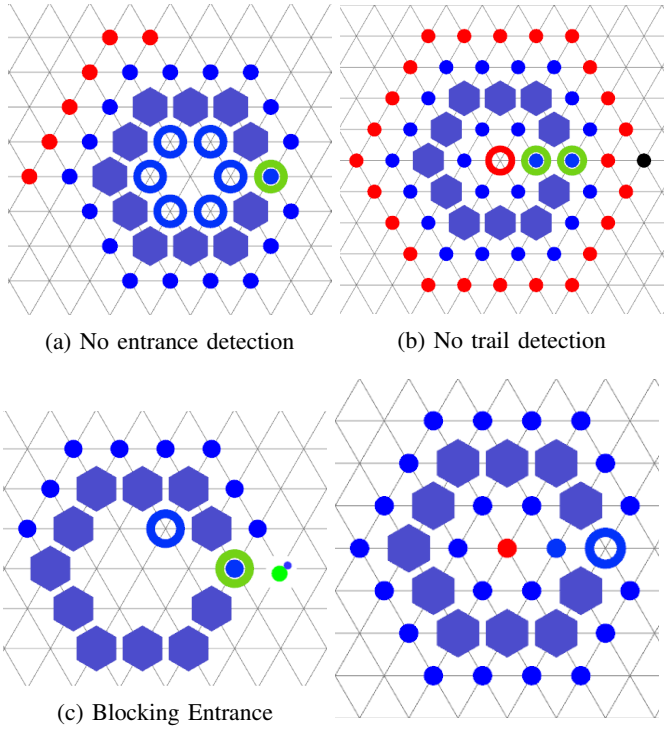
(a) Legal (b) Ideal (c) Layered

Figure 1: Valid Coatings



(a) Cave shaped object (b) Trail entrance- (d_e) and subjects-diameter (d_s)

Figure 2: A Cave-shaped Object Formation and its Diameter



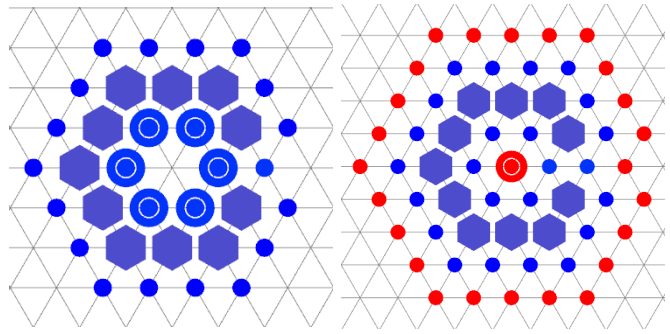
(a) No entrance detection (b) No trail detection

(c) Blocking Entrance (d) No amount control

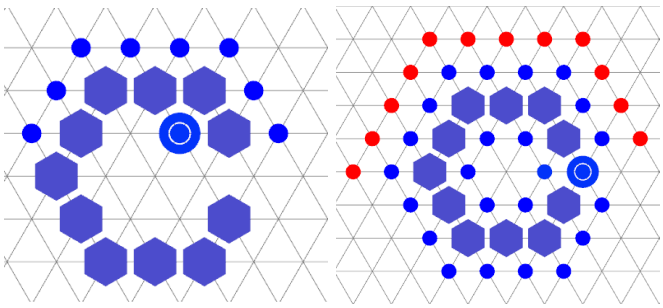
Figure 3: Cave Problems

Following that, the red subjects coated should be coated in the blue marked positioned, so an valid coating happens such as shown in Fig. 4a. Thus, the challenge here is to make the subjects capable of recognizing a cave entrance while scanning.

Challenge 2. *Detecting the trail.* Assuming the cave entrance is found the next challenge is the trail blocking. This challenge is illustrated in Fig. 3b. We can see the three different subject colors. Each color represents the coated layer level. Blue subjects denote first layer of coating, red being the second layer, and black for the third layer. The blue subject marked with green are seen to be blocking the trail that leads to the core of the cave marked with a red circle as it should be a second layer location. Consequently, an invalid coating has occurred as the black subject should ideally be at that location



(a) Entrance detection (b) Trail detection



(c) No blocked entrance (d) Amount Control

Figure 4: Valid Coated Caves

as shown in Fig. 4b. Therefore, the subjects need to know somehow while scanning that there is a trail that should not be coated until all required positions inside the cave are covered.

Challenge 3. *Order of coating.* The next challenge is how to position the subject during the coating process so that the trail entrance is not blocked to early as in Fig. 3c. Even when knowing the trail position, the number of subjects and locations around the object shape to occupy by subjects, the order in which the locations as occupied is relevant. When the trail entrance is blocked by a subject to early, as here marked with a green circle, further positions cannot be occupied anymore. Taking the order to positioning into account, i.e. not blocking the trail entrance, would leak to successful coating, as shown in Fig. 4c.

Challenge 4. *Awareness of the number of subjects and positions.* The last challenge to overcome is to be aware of the number of subjects and the positions available around the object to be coated. If the number of subjects is exactly sufficient to fill up a complete layer, the core of the cave should not be coated otherwise an invalid coating occurs. This kind of challenge can be seen in Fig. 3d. In this scenario the subjects are exactly enough to coat the first layer of the cave. However, the red particle is coated in the middle of the cave, but this location is actually in the second layer. Thus, the red subject is at the wrong location and should be actually at the position marked with a blue circle as shown in Fig. 4d. In this challenge, the task is to be aware of the subjects left against the remaining scanned free location so as to decide how to coat the subject to prevent an invalid coating from occurring.

These four main challenges for the coating of cave objects are most prominently to overcome in shaping a solution for swarm-based coating. Next, we introduce the terminology used throughout the algorithm descriptions are defined.

III. Terminology

In this chapter first the necessary terminology for the coating is defined. In addition we define the ability of the subjects and especially of the leader, at last we describe how a valid coating can be measured.

Subject. A subject is an active matter that has a constant diameter d_s and has the capability of doing activities autonomously, such as scanning its surroundings, moving, as well as picking and carrying other subjects. Examples of active matters are robots, animals, software programs or even humans. We define S as the set of *subjects* s.t. $S = \{s \mid s \in \mathbb{N}_+\}$.

Objects. An object, unlike the subject, is passive and cannot do any action on its own. Objects can be for example neoplasm, cancer cells or walls. We define O to be the set of objects s.t. $O = \{o \mid o \in \mathbb{N}_+\}$, and $S \cap O = \varnothing$.

Matter. A matter is the combination of subjects and objects.

$$M = \{m \mid m \in \mathbb{N}_+\}$$

It is a partition M: $S \cup O = M$. These matters are located on Locations.

Location. A location l is an point on which a *Matter* is placed. As the coating algorithm is designed for two dimensional (2D) spaces, the location l is a 2D vector s.t. $l = (x, y) \wedge x, y \in \mathbb{Z}$. We then define L to be the set of all locations that build up together the two-dimensional plane area, which allows to define proximity.

Each element, either an object or a subject, of a *Matter* is positioned on one of the *locations*. The function for the mapping is defined as: $\lambda : M \to L$ and $\lambda(m) = l$. λ is injective, meaning that, only one *Matter* can be on one location at any time.

Any *location* that is not occupied by any matter is called a *free location*. Let FL be the set of free *locations* s.t. $FL = \{fl \mid fl \in L \setminus \lambda(m)_{\forall m \in M}\}$. Additionally, a location can be occupied by either a *subject* or an *object*. Let SL be the set containing all the locations occupied by *subjects* s.t $SL = \{sl \mid sl \in L \setminus (FL \cup \lambda(o)_{\forall o \in O})\}$ Also Let OL be the set holding all the locations occupied by *objects* s.t. $OL = \{ol \mid ol \in L \setminus (FL \cup SL)\}$. A *subject* can also carry another *subject*meaning that a relation between two *subjects* can exist. We define his relationship as $SS = S \times S$. Thus, a composition between $SS \circ SL$ is also possible.

Neighbor. Each *location* is connected to a maximum of $\hat{n}$ adjacent *locations*, termed Neighbors. Let N_l be an index set of neighbor locations for every *location* l s.t. $N_l = \{n_j \mid l \in L, n_j \in L \setminus l, 1 \leq j \leq \hat{n}\}$. We assume a uni-distance model, in which the distance between each *location* is constant (1) and larger than the diameter of one *subject* and lower than the combined diameter of two *subjects*.

Link. A link is the connection between two *locations*. Let l_a and l_b be two locations. There exists a *Link* LI (l_a, l_b) if $l_a \in L$ and $l_b \in N_{l_a}$

Direction. This is a numbering of the order the *Links* relative to a specific *location*. The direction of the *Links* relative to a *location* is given as $DIR = \{dir_i \mid 1 \leq i \leq \hat{n}\}$, and each direction is an 2D vector given as $dir = (x, y) \mid x, y \in \mathbb{Z}$.

For the coating algorithm, we can define two important functions:

- $moveTo(l, dir_i)$: Returns the ith adjacent location for given location l.
- $getDirForLocation(l)$: Returns the direction for the adjacent location.

Path. This is the set of *Links* through which two *locations* are connected. We define $Paths(l_p, l_q)$ for two *locations* $l_p, l_q \in L$ to be the set of possible *Links* LI $\{(l_p, l_1), (l_1, l_2), ..., (l_{q-1}, l_q)\} \subseteq LI$ through which, l_p and l_q are connected. It is important to note that several possible paths may exist and the shortest path is preferred.

For that the set of **Shortest Paths** is defined as

$$minPaths : L \times L \rightarrow \mathcal{P}(LI)$$

$$p \in minPaths(l_p, l_q) \subseteq Paths(l_p, l_q)$$

$$when\ p \in Paths(l_p, l_q) \wedge \nexists\ q \in Paths(l_p, l_q) : |q| < |p|$$

Distance. Based on the path, it is necessary to define the **Distance** between two *locations*. Generally speaking, the *Distance* between two *locations* is the length of the *shortest path* between these two *locations*. The *Distance* $D : L \times L \rightarrow \mathbb{N}_+$ is defined as the minimum number of *Links* between the two *locations* $D(l_p, l_q) = |p|\ with\ p \in minPaths(l_p, l_q)$.

Distance to the Object Shape. Based on the distance between any two locations, we define next the Distance to the Object Shape ($DOS_O : L \rightarrow \mathbb{N}_+$). It describes the distance from a given location to the nearest location that is occupied by an object and is defined as

$$DOS_O(l_p) = |p|\ with\ p \in minPaths(l_p, \lambda(o \in O)).$$

Hop. This is measure of distance between two *subjects*. It is the distance $D(\lambda(s_1), \lambda(s_2))$ between the locations of two *subjects* s_1 and s_2.

Coating Validity. A coating is valid if the following is fulfilled:

$$max(DOS_O(sl)_{\forall sl \in SL}) \leq min(DOS_O(fl)_{\forall fl \in FL}) \quad (1)$$

As a result, a coating algorithm is valid if and only if the maximum distance between any subjects locations and the object shape O is less or equal to the minimum distance between any free location and the object shape O, i.e. the subjects occupy all closest locations around the object shape.

IV. Related Work

Before introducing our leader-based coating approach, we discuss the related work for this problem statement. The most recent one is [1] and [2]. They have defined the coating problem and proposed a coating algorithm termed "universal coating algorithm". This algorithm works for a swarm of particles (subjects) that relate to each other as chain and starts to coat the object by watching who is still not in a coating position within the chain. However, as they have pointed out in their paper, their algorithm is not working for objects with tunnels and caves. Further papers address similar problems as the coating problem, but not its central challenges. [3], [4] and [5] are focused on caging an object and moving it away. With caging it is not even meant to cover the whole object shape. This could be fatal if the objects are flexible enough to uses tiny gaps for escaping. Formation algorithms such as in [6], [7] and [8] aim to move a swarm in a coordinated fashion, often to a desired shape. Obstacles are not considered. Solutions in [9] and [10] are scanning the object with laser or radiation, calculating finger points so the a robot can grasp it. A few points are sufficient. Here in contrast, we aim to use all available subjects for coating. Nevertheless, another problem statement is close to coating termed target surrounding or boundary coverage. For example in [11], [12] and [13], solutions are presented. They are using similar ways of coating, but the subjects have higher capabilities and all of them are focused on objects that have a simple shape. To the best of our knowledge there exist no solutions that are capable to coat object shapes with caves as presented here.

V. Our Leader-based Coating Algorithm

In the main idea of our Leader-based Coating Algorithm, we define a specific subject in the swarm as leader. Its main task is to scan the object shape, identify a potential cave entry, enter the cave and memorize the available places, and after scanning of the object shape to go and pick a subject in the swarm as long as there are any left and place it on a suitable next position in the coating. During this procedure, the leader must take care to not place subjects on the trail inside the cave. At the end it positions itself. For the definition of the algorithm, we first define a few further terms that will be used in the description of our algorithm. In specific the new terms allow to describe the surrounding and environment of the the subjects and especially the leader. The presentation of our algorithm begins then in Subsection V-B and elaborates in Subsection V-C how we address the specific challenges stated in Section II.

A. Additional Terminology

State. A location can have two states: occupied or free. Let $mIn(dir)$ be a function that return true if an adjacent location is occupied by an matter in direction dir. Thus, the neighbors (adjacent locations) of a location l can have different states. For the coating algorithm the neighbors states are getting labeled based on the viewing direction to an adjacent location and the left and right side location of the viewing direction. As a result, the neighbors needs to be labeled.

Labels. They are five types of labels: Previous (pr), trail (tr), total free (tf), free (fr), and occupied (oc). First, a viewing location will be labeled as "previous" $pr \in \mathbb{N}_+$ Next, a trail $tr \in \mathbb{N}_+$ label is when the adjacent location is free and the left and the right side is occupied. Please note that it is sufficient to keep a trail of one location width when coating. Third, total free $tf \in \mathbb{N}_+$ is when it is free and additionally the left and right adjacent location of l are free, too. Fourth, free $fr \in \mathbb{N}_+$ is when one of the left or the right adjacent location is occupied, and the other side is free.

Finally, an adjacent location is labeled as occupied $oc \in \mathbb{N}_+$ when a matter is on it. The values of the labels must be different and much lower than the previous one as shown $pr \ll oc \ll fr \ll tr \ll tf$. This is necessary for knowing the environment which will be explained in the next chapter. Let label $LA = \{pr, tr, tf, fr, oc\}$ be a set that contains all these labels.

Neighborhood Array. Let environment NH be an array that contains tuples. Each tuple consist of an adjacent location label and its specified direction. Let $NH = [(la, dir)_i]$ with $la \in LA, dir \in Dir, 0 < i \leq \hat{n}$ and $NH(la) = dir$ a function that returns for a label $la \in LA$ its direction.

Environment. Summing up all the labels of the neighbors gives information about the environment. This helps later to find a trail entrance, the inside and outside of a cave, and a dead end. Fig. 5 shows label types. The different colored lines are representing the labels. The green line with the arrow is showing that the adjacent location is the previous visited location pr. The arrow stands from the direction that the subject came from. The blue lines are labeling the occupied adjacent location oc. The red line shows an adjacent trail entrance tr. And finally, the orange one is a free location fr that is beside a matter/object. The fr is necessary for scanning and moving around an object.

Leader. In the Leader-based Coating Algorithm a swarm of subjects are searching for an object to coat. If they found the object a leader is elected of them. How the leader is elected is not relevant, various leader election algorithms are available and can be used. What is important, is that this leader-subject has some features. The Leader knows its own location $own_l \in L$, the previous location $prev_l$ that it has been before, the location of an object $u_l \in OL$, and the locations of the other subjects SL. This knowledge can be obtained by searching the space in form of a swarm, locating one single object of the connected object shape and then broadcasting the information on $u_l \in OL$ as well as the subject locations in the swarm.

Further, the leader can has a memory and can store information on unoccupied locations $U_L = [l_1 ... l_n] | l_n \in FL$ and trail locations $T_L = [l_1 ... l_n] | l_n \in FL$ as an array. The array is a stack and a queue. So, in order to insert a value in front of the array the function the function $insert(array, value)$ will do it. Next, to insert a value at the end of array the $push(array, value)$ function can be used. With $extent(array1, array2)$ the $array1$ will be extent at its tail by $array2$. The function $shif(array)$ will return the first $value$ of the $array$ and delete it from the array. The inverse is $pop(array)$ that takes the last $value$ of the $array$ and delete it from the $array$. It has to be mentioned that the leader cannot calculate its distance towards to any matter and can only calculate its direction towards them. Finally, a leader can take and drop a subject with the function $takeSubjectFrom(location)$ and $dropSujectOn(location)$.

B. Algorithm Details

The needed terminologies are defined, and the problems specified. In this section we elaborate our coating algorithm.

Leader-Based Coating Algorithm (Alg. 1). A brief overview of how the leader-based coating algorithm works is shown in

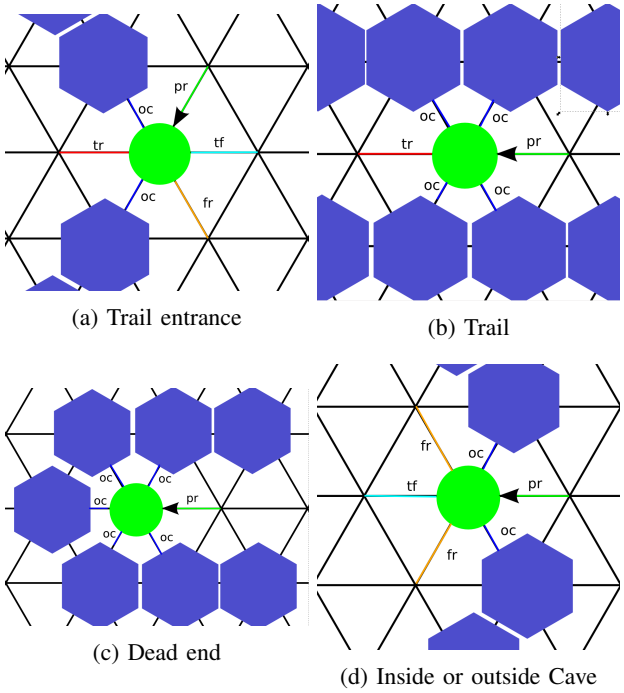


Figure 5: Environment and Neighborhood Types

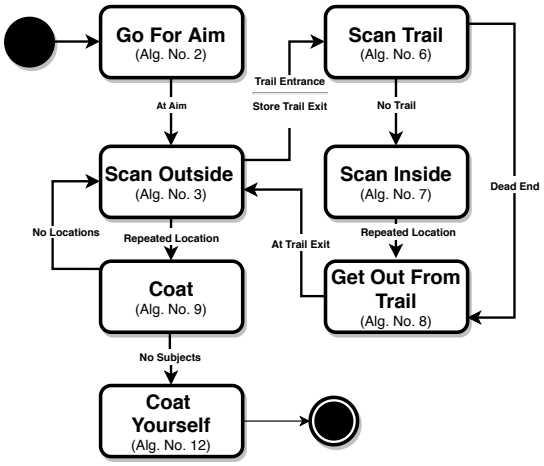


Figure 6: Overview: Leader Based Coating Algorithm

Fig. 6 and the algorithm in Alg. 1. The steps are described next.

Go For Aim (Alg. 2). Before the coating can be started, the leader needs to go to the object to scan it. Assuming it knows the location of one single object (aim_i) and its own location, it identifies in which direction the object lies and walks in that direction.

Scan Outside (Alg. 3). After it reached the object, it starts to scan the outside shape. It memorizes each location in the array U_L and moves to the next location until its own location repeats in U_L Indeed, for a simple object the outside scanning would be enough to identify all locations of distance 1 to the object shape. These locations could be coated and then the process could start again identifying the already coating subjects as elements to coat. However, this algorithm should work for cave

Algorithm 1 Leader Based Coating Algorithm

```
procedure LBCA(u_l, SL)
    own_l ← goFor(u_l)
    while SL do                                  ▷ They are subject locations
        U_L ← scanOutside()
        coat(U_L, SL)
    coatYourself(U_L)
    Finished Coating
```

Algorithm 2 Go For Aim

```
procedure goFor(aim_l)
    SP ← minPath(own_l, aim_l)
    nextLocation ← shift(SP)
    while nextLocation ≠ aim_l do
        nextDir ← direction to nextLocation
        prev_l ← own_l
        own_l ← moveTo(nextDir)
        nextLocation ← shift(SP)
```

objects. Therefore, the leader scans after each movement its environment.

Scan Trail(Alg. 4) . If the leader recognizes a trail entrance, it will enter and solves the first challenge of finding a trail entrance 3a. In order to solve the second challenge 3b of not blocking the trail, the trail locations are stored separately in the T_L array. The trail location will be coated after finishing this layer, before the scanning for the next layer starts in order to get a valid coating as shown in Fig. 4b. The trail scanning stops either at a dead end or the inside of the cave has been reached.

Get Out From Trail(Alg. 5). When a leader reaches a dead end, it gets out from the trail. The outside scanning continues. For the coating the trail locations will be used immediately which would not happen if the trailed ended with a cave.

Scan Inside (Alg. 6). Inside the cave, the leader starts to scan it. It is doing almost the same like the outside scan but with a tiny difference, the environment scanning is off as no cave entry is to be found (again). The inside scanning stops when the leader location is repeated in the trail location array T_L. Then outside scanning continues until it reaches the same location that has been scanned earlier. After that the coating starts based on the two arrays in the memory of the leader.

Coat(Alg. 7). In the coating process the leader takes the other subjects one by one and positions them on the scanned locations stored in its memory. The process of coating this current layer finishes when there are no scanned locations. If subjects are left, the leader starts again the outside scanning process for the next layer.

Coat Yourself(Alg. 8). If no subjects are left, it is time that leader selects its own position in the coating.

This was a brief overview of the leader-based coating algorithm. Next, we present our solutions for each specified challenge in Section II. With help of pseudo codes the solutions are explained in depth.

C. Addressing the Challenges stated by Caves

Solution 1. *Recognizing a trail entrance.* For the solving the entrance challenge the important approach is the scanning of the environment, shown in Algorithm 9. The

Algorithm 3 Scan Outside

```
1:  procedure scanOutside
2:      deadEnd ← False
3:      T_L ← [emptyArray]
4:      U_L ← [emptyArray]
5:      while own_l ∉ U_L do
6:          ENV, NH ← scanEnvironment()
7:          if ENV = TRAIL_ENTRANCE then
8:              trailExit ← own_l
9:              T_L, ENV, NH ← scanTrail(NH(tr))
10:             if ENV = DEAD_END then
11:                 deadEnd ← True
12:             else
13:                 U_L ← extend(U_L, scanInside(T_L))
14:             getOut(trailExit)
15:             ENV, NH ← scanEnvironment()
16:         else
17:             push(U_L, own_l)
18:         prev_l ← own_l
19:         own_l ← moveTo(ENV(fr))
20:     if |SL| < |T_L| + |U_L| then
21:         reduceScannedLocations(SL, T_L, U_L)
22:     else if deadEnd then
23:         U_L ← extend(U_L, T_L)
24:     return U_L
```

Algorithm 4 Scan Trail

```
procedure scanTrail(trDir)
    T_L ← [emptyArray]
    push(T_L, own_l)
    prev_l ← own_l
    own_l ← moveTo(trDir)
    push(T_L, own_l)
    ENV, NH ← scanEnvironment()
    while ENV ≠ DEAD_END ∧ ENV < tf do
        prev_l ← own_l
        own_l ← moveTo(NH(tr))
        push(T_L, own_l)
        ENV, NH ← scanEnvironment()
    return T_L, ENV, NH
```

Algorithm 5 Get Out From Trail

```
procedure getOut(exit)
    own_l ← goFor(exit)
    nextDir ← getDirForLocation(exit)
    prev_l ← own_l
    own_l ← moveTo(nextDir)
```

Algorithm 6 Scan Inside

```
procedure scanInside(NH, T_L)
    SC ← [emptyArray]
    prev_l ← own_l
    own_l ← moveTo(NH(fr))
    while own_l ∉ T_L do
        ENV, NH ← scanEnvironment()
        push(SC, own_l)
        prev_l ← own_l
        own_l ← moveTo(NH(fr))
    return SC
```

Algorithm 7 Coat

```
procedure coat(U_L, SL)
    while U_L ∧ SL do
        l_s ← pop(SL)
        goFor(l_s)
        takeSubjectFrom(l_s)
        u_l ← pop(U_L)
        goFor(u_l)
        dropSubjectOn(u_l)
```

scanEnvironment function scans the adjacent locations and label them as shown in Alg. 9 line 8 and 12 while using

Algorithm 8 Coat Yourself

```
procedure coatYourself(S_L, T_L)
    if U_l = [emptyArray] then
        U_L ← scanOutside()
    u_l ← pop(U_L)
    SP ← minPath(own_l, u_l)
    while own_l ≠ u_l do
        nextDir ← getDirForLocation(pop(SP))
        own_l ← moveTo(nextDir)
```

Algorithm 10. These labels are getting summed up in order to get a value for the environment that is ENV. With these values the leader knows in what kind of environment it is. That is the reason why the value of the labels have to be different and vary according to magnitudes. The reason for that is that the sum of the labels should not interfere any of the label values. Fig. 5 shows the different types of environment. For example the trail entrance tr is constant and named as $TRAIL_ENTRANCE$ (Fig. 5a). For a trail there is only one tr, one pr, and the rest oc. A dead end is the sum of $5*oc+pr$ label the constant name is $DEAD_END$ (Fig. 5c). If the environment has at least one tf and no tr, either it is an inside or outside cave environment. Basically, the leader scans at the beginning and after each movement in the $scanOutside$ (Alg. 3) the environment and if the environment type is $TRAIL_ENTRANCE$ it starts to scan the trail, which leads for solving the second challenge as shown in Fig. 3b.

Solution 2. *Detecting the trail.* Before scanning the trail, it is important for the leader to store its own position as $trailExit$ for using it later of getting out of the cave and trail. In algorithm 4 the leader stores the entrance location in the array T_L, it moves then towards the trail and stores every location until it reaches a $DEAD_END$ or inside the cave, that is the environment has a tf. This separation of trail location T_L from the uncoated locations U_L solves the challenge in Fig. 3b and 3c. Because the leader will consider first uncoated (and non-trail) locations while coating. However, if the leader reaches a $DEAD_END$ while it is in a trail, the trail locations will be extended to uncoated locations as shown in Alg. 3 line 23.

Solution 3. *Order of coating - not blocking the trail entrance.* Nevertheless, without the trail scanning the coating algorithm will face the challenge as shown in Fig. 3c while coating. The reason is the way of reading the stored location. If leader reads the array with the first in first out (fifo) criteria, the entrance of the trail will be blocked. Thus, this can be solved by just using a last in first out (lifo) memory read, as with this the leader will reverse its array and start from the end of the array list and therefore coat the entrance last, as it was stored before the cave/trail were scanned.

Solution 4. *Awareness of the number of subjects and positions.* Up to now, three out of four challenges are addressed. The last challenge is reacting on the number of subjects and objects, as shown in Fig. 3d. The problem here is that if there are not enough subjects for coating the entire scanned area, the leader has to reduce the scanned locations in such a way so the coating criteria can be fulfilled. As a result, after each scanning the leader has to check if there are enough subjects for coating he entire scanned locations (see Alg. 3 line 20) and react on this information. By not having enough subjects, the leader needs to reduce its scanned locations. The reduction happens

Algorithm 9 Overcoming Challenge 1: Scan Environment

```
1:  procedure scanEnvironment
2:      NH ← [emptyarray]
3:      ENV ← 0
4:      for i ←, n̂ do                                   ▷ here n̂ := 6
5:          prevDir ← getDirectionFor(prev_l)
6:          viDir ← dir_i
7:          if prevDir ← viewDir then l
8:              labelType ← prev
9:          else
10:             leDir ← i − 1 mod |DIR|
11:             ritDir ← i + 1 mod |DIR|
12:             labelType ← getLabel(viDir, leDir, riDir)
13:             push(NH, (viewDir, labelType))
14:             ENV ← ENV + labelType
15:     return ENV, NH
```

Algorithm 10 Overcoming Challenge 1: Get Label

```
procedure getLabel(ViewDir, leftDir, rightDir)
    if no mIn(ViewDir) then
        if mIn(leftDir) ∧ mIn(rightDir) then
            return tr
        else if mIn(leftDir)∧ no mIn(rightDir) then
            return fr
        else if mIn(rightDir)∧ no mIn(leftDir) then
            return fr
        else if no mIn(rightDir)∧ no mIn(leftDir) then
            return tf
    else
        return oc
```

Algorithm 11 Overcoming Challenge 4: Reduce Scanned Locations

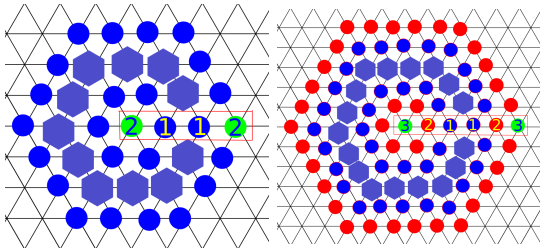
```
1:  procedure reduceScannedLocations(S_L, U_L, T_L, deadEnd)
2:      if |S_L| < |T_L| then
3:          if not deadEnd then
4:              pop(T_L)
5:          U_L ← reduceTrailLocations(T_L, S_L)
6:      else
7:          for i=0 do|S_L| − |T_L|
8:              shift(U_L)
9:          U_L ← extend(U_L, T_L)
10:     return U_L
```

Algorithm 12 Overcoming Challenge 4: Reduce Trail Locations

```
1:  procedure reduceTrailLocations(S_L, T_L)
2:      i ← 0
3:      while |T_L| > |S_L| ∧ T_L ≠ [emptyarray] do
4:          if counter mod 2 = 0 then
5:              pop(T_L)}
6:          else
7:              shift(T_L)
8:          i ← i + 1
9:      return T_L
```

in Alg. 11. As we can see there are two types of reductions. First, if the amount of subjects is not enough to coat the trail (alg. 11 line 2) and second if there are enough subjects to coat the trail but not enough to coat the entire scanned locations for the current layer (alg. 11 line 7).

Assuming that, there are not enough subjects to coat the trail, first it has to be checked if the trail has a dead end. If not so, the last entry of the trail has to be deleted. The reason for that is that the scanned trail locations includes the trail entrance and the first location in the cave. This is so, because to prevent to block either the entrance of the trail or the cave core. And both locations distance are always equal to the actual level of coating layer. As a result, the trail consist of a head and a



(a) Head and tail with distance 2

(b) Head and tail with distance 3

Figure 7: Examples of trails with their distances

tail that have the highest distance and the distances are getting less each time the head and tail is cut. Examples of trails are shown in Fig. 7. Thus, depending on the length of the trail array, the cave entry locations have the distance one and from both sides the distances are increasing with the same amount. Accordingly, having not enough subjects, it is important to fill up the trail locations with lower distances. Thus, trail locations have to be reduced from the head and the trail. This is the reason why in Alg. 12 line 4 is checking for an even number or odd number. The reason is that it is swapping between deleting the head and next in the trail to coat the ones with the lower distance. For this purpose, the trail locations have a higher priority than the inside and outside scanned locations, because it includes locations with lower distances compare to the actual coating distance.

Next, in case that they are enough subjects for covering the trail but not enough for coating the rest of the layer, the uncoated locations array has to be reduced by the rest of the number of subjects as shown in Alg. 11 line 7. With the reduction of the scanned location a validated coating can be guaranteed as shown in Fig. 4d.

Now that solutions for the problems have been shown, the next chapter proves empirically that our Leader-based Coating Algorithm manages to coat cave-like object shapes correctly.

VI. Proof of Concept and Evaluation

For the proof of concept of the leader-based coating algorithm the simulator **Swarm Sim** [14] has been used. This simulator is created at the "Technology of Social Networks Lab" of the Heinrich Heine University in Düsseldorf/Germany. It is a 2D and 3D simulation core that provides interfaces for creating individual scenarios and solutions with the main focus of simulating swarms. This simulator is open source and can be downloaded from [15], the leader-based coating algorithm is available from [16]. The proof of concept of the leader-based coating algorithm is presented with this simulator.

For the proof of concept, the cave as shown in Fig. 2a is used with subjects numbers starting from 1 up to 114. The reason for using 114 subjects is that with this amount the inside and four layers around the cave can be coated. After each coating has been finished the simulator checks if the coating is valid. The success rate is the ratio of valid coatings for subjects counts from 1 to 114.

Proof 1. The first proof of concept starts with the simple coating algorithm (Fig. 8a). Here the leader is not checking for cave entrance and thus the invalid coating starts after the outside layer of the cave has been coated, the second layer just started and the inside of the cave is left behind. As a result, the success rate is 15.79%.

Proof 2. Next, the leader can recognize the cave entrance and uses lifo to fill the cave up from its innermost valid position. Using lifo and entrance the success rate is almost three times (Fig. 8b) more than without checking for the entrance. However, the failure of coating begins after the second layer has been coated and the third layer begins. The innermost position in the cave has a distance of 2 to the objects and will not be coated. Consequently, this is the reason for only a 42.11% success rate.

Proof 3. By adding the trail scanning the success rate increases to 98.25% (Fig. 8c). The missing 1.75% is because this algorithm does not check the amount of left subjects and causes that the inside of the cave is filled up even though a location with a lower distance outside of the cave remains uncoated.

Proof 4. Finally the last element of the algorithm is added and evaluation: the reduction of the arrays of positions to be coated due to awareness of the number of subjects and positions to be covered. This completed Leader-based Coating Algorithm is tested with various of caves of different core sizes and position of entrances. For all of them, the success rate was 100% as shown in Fig. 8d. Thus, we can conclude that with the presented algorithm for leader-based coating all caves are successfully covered.

We have shown that cave-shaped objects are successfully coated with our approach.

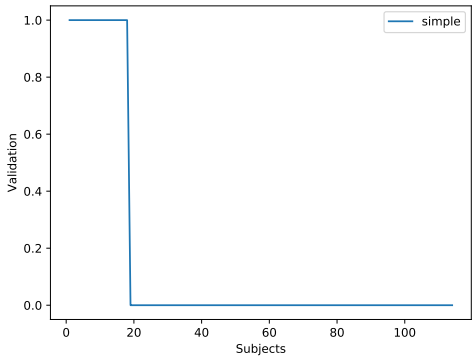
VII. Conclusion and Future Work

We introduced the problem of coating of cave-shaped objects and the terminology used in this context. Such non-convex object shapes might occur in form of cancer cells that are to be coated by future nanobots. We introduced in this paper a Leader-based Coating Algorithm that is capable to coat such structures successfully. We derived from a simple algorithm the essential elements to address the challenges needed to coat cave-like structures. With a step by step description of the added functionalities, we presented the algorithm to coat a cave-shaped object independent of the amount of the subjects with 100% success rate.

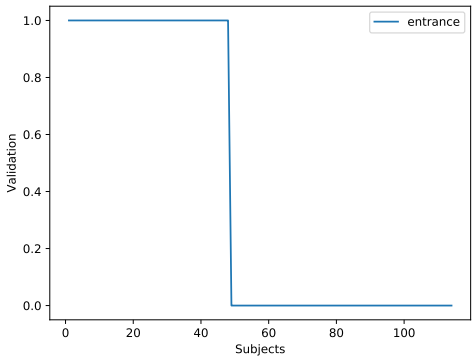
For future work, we aim to address the coating of object shapes with nested caves (cave in cave) and objects that have many caves or even a maze. Additionally, as pointed out, we assume object shapes with entries of the size of one subject. Shapes with bigger entries first need to be coated layer by layer to reach that entry size. It would be advantageous to include this process in a generalized algorithm. Finally, we aim to extend this algorithm to 3 dimensional structures. The simulator swarm sim supports the programming of such algorithms as well. In reality, cancer cells are 3 dimensional as well.

Acknowledgments

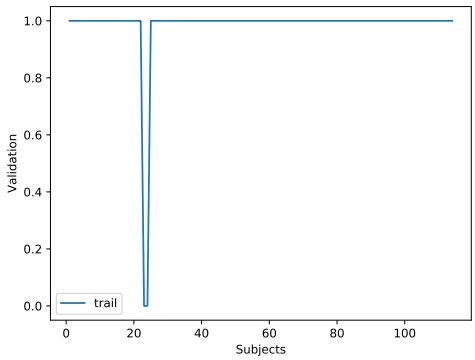
We would like to thank the Arab-German Young Academy of Sciences and Humanities for funding this research as well



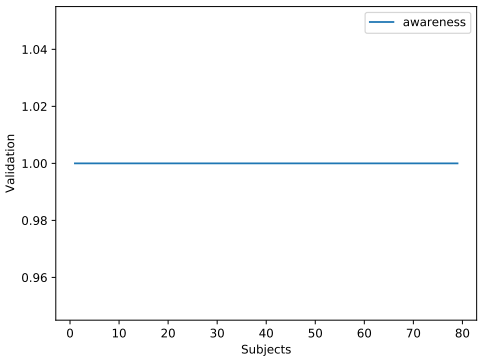
(a) Simple: Success rate 15.79%



(b) Entrance and lifo: Success rate 42.11%



(c) Trail scanning: Success rate 98.25%



(d) Reduction: Success rate 100%

Figure 8: Success Rates of the various Algorithm Elements

as our collaboration partner Prof. Dr. Ahmed Khalil for the valuable and inspiring discussions.

References

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

[2] J. J. Daymude, Z. Derakhshandeh, R. Gmyr, A. Porter, A. W. Richa, C. Scheideler, and T. Strothmann, "On the runtime of universal coating for programmable matter," *Natural Computing*, vol. 17, no. 1, pp. 81–96, 2018.

[3] W. Wan, F. Lu, and R. Fukui, "Error-tolerant manipulation by caging," *Signal Processing*, vol. 120, pp. 721–730, 2016.

[4] S. Wilson, T. P. Pavlic, G. P. Kumar, A. Buffin, S. C. Pratt, and S. Berman, "Design of ant-inspired stochastic control policies for collective transport by robotic swarms," *Swarm Intelligence*, vol. 8, no. 4, pp. 303–327, 2014.

[5] Z. Wang and V. Kumar, "Object closure and manipulation by multiple cooperating mobile robots," in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, vol. 1. IEEE, 2002, pp. 394–399.

[6] S. Mahmood, M. Liaquat, Z. Ahmad, A. Zaheer, and A. Khan, "Study of Formation Control of Mobile Robots," *Int. Journal of Mech. Eng. and Robotics Research*, vol. 9, no. 1, pp. 111–116, 2020.

[7] A. Marzoughi, "Switching Navigation for a Fleet of Mobile Robots in Multi-Obstacle Regions," *Int. Journal of Mech. Eng. and Robotics Research*, vol. 8, no. 1, 2019.

[8] M. A. Alshawi and M. B. Shalan, "Minimal Time Dynamic Task Allocation for a Swarm of Robots," *Int. Journal of Mech. Eng. and Robotics Research*, vol. 6, no. 6, pp. 481–487, 2017.

[9] Q. Lei and M. Wisse, "Object grasping by combining caging and force closure," in *2016 14th International Conference on Control, Automation, Robotics and Vision (ICARCV)*. IEEE, 2016, pp. 1–8.

[10] J. Liu, S. Xin, Z. Gaol, K. Xu, C. Tu, and B. Chen, "Caging loops in shape embedding space: theory and computation," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–5.

[11] G. P. Kumar and S. Berman, "Statistical analysis of stochastic multi-robot boundary coverage," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 74–81.

[12] T. P. Pavlic, S. Wilson, G. P. Kumar, and S. Berman, "An enzyme-inspired approach to stochastic allocation of robotic swarms around boundaries," in *Robotics Research*. Springer, 2016, pp. 631–647.

[13] A. A. Semakova, K. S. Ovchinnikov, and A. S. Matveev, "Self-deployment of mobile robotic networks: an algorithm for decentralized sweep boundary coverage," *Robotica*, vol. 35, no. 9, pp. 1816–1844, 2017.

[14] A. R. Cheraghi, K. Actun, S. Shahzad, and K. Graffi, "Swarm-Sim: A 2D & 3D Simulation Core for Swarm Agents," in *3rd Int. Conf. of Intelligent Robotic and Control Engineering (IRCE'20)*, 2020.

[15] A. R. Cheraghi and U. Duesseldorf. (2020) Swarm simulator. [Online]. Available: https://gitlab.cs.uni-duesseldorf.de/cheraghi/swarm-sim

[16] A. R. Cheraghi. (2020) Leader based coating algorithm. [Online]. Available: https://gitlab.cs.uni-duesseldorf.de/cheraghi/swarm-sim/-/tree/leader_coating