Scattering with Programmable Matter*

Alfredo Navarra[1], Giuseppe Prencipe[2], Samuele Bonini[2], Mirco Tracolli[1]

[1] Dept of Mathematics and Computer Science, University of Perugia, Perugia, Italy.
alfredo.navarra@unipg.it; mirco.tracolli@pg.infn.it
[2] Dept of Computer Science, University of Pisa, Pisa, Italy.
giuseppe.prencipe@unipi.it; sbonini7@studenti.unipi.it

Abstract. We aim at studying the *Scattering* problem (or *Distancing*) in the context of *Programmable Matter* (PM). This is intended as some kind of matter with the ability to change its physical properties (e.g., shape or color) in a programmable way. PM can be implemented by assembling a system of self-organizing computational entities, called *particles*, that can be programmed via distributed algorithms. A rather weak model proposed in the literature for PM is SILBOT, where particles operate in the nodes of a triangular grid. A particle can be *contracted*, occupying one node, or *expanded*, occupying one node and one adjacent edge. The movement happens in two steps: a particle expands towards a node and, if empty, it moves there in the contracted state. Particles are all identical, executing the same algorithm based on their local neighborhood. They have no direct means of communication and are disoriented. We aim to let particles move so as to achieve Scattering, i.e., all particles are at least two hops far apart from each other. We show that the problem is unsolvable within the pure asynchronous setting where the activation of each particle is left to an adversarial scheduler. Then, we consider the case where contracted particles react to the stimuli of neighboring particles, i.e., a particle is activated as soon as it is neighboring with another one. By this event-driven variant, we are able to provide a resolution algorithm along with its correctness. Furthermore, we investigate (also by simulations) on configurations where some nodes of the grid can be occupied by immovable elements (i.e., obstacles).

1 Introduction

The rise in the last decade of new capabilities to design smart systems, compute and fabricate like never before, has sparked a renewed interest in the performance of materials. In particular, we are now witnessing significant advances in studies dedicated to "active matter", i.e., 3D/4D printing, materials science, synthetic biology, DNA nanotechnology and soft robotics, which have led to the development of more and more technologies that rely on smart combinations of materials know-how, software, hardware and to the growth of the new field called *Programmable Matter* (PM). This refers to some kind of matter with the ability

* The work has been supported in part by the Italian National Group for Scientific Computation (GNCS-INdAM).

to change its physical properties (e.g., shape or color) in a programmable way. PM can be implemented by assembling a system of self-organizing computational entities, called *particles*, that can be programmed via distributed algorithms to collectively achieve global tasks.

Several theoretical models for PM have been proposed, ranging from DNA self-assembly systems, (e.g., [20, 22]) to metamorphic robots, (e.g., [23, 19]), to nature-inspired synthetic insects and microorganisms (e.g., [13]), each model assigning special capabilities and constraints to the particles and focusing on specific applications. Among them, the Amoebot model [8–10, 12] is of particular and immediate interest from the distributed computing viewpoint. Indeed, in such a model (introduced in [10], and so called because it is inspired by the behavior of the amoeba), PM is viewed as a swarm of decentralized autonomous self-organizing particles (represented by finite automata) that form a connected structure with the help of local bonds.

The capabilities associated with the particles have been further reduced in a recent model called SILBOT [7]. In particular, in SILBOT, a triangular grid is considered where particles move to accomplish specific tasks. The movement of a particle is obtained in two steps: first a particle expands towards a neighboring node along a joining edge, and successively it reaches the desired node, if empty. Basically, each particle may assume two different states: CONTRACTED, i.e., a particle occupies one node; EXPANDED, i.e., a particle occupies one node and an adjacent edge. We consider a classical problem in distributed computing that is called *Scattering* (see, e.g. [3, 18]): starting from any configuration where particles are all CONTRACTED, the aim is to lead the system to a configuration where each particle is still CONTRACTED but admits no neighboring particles.

2 The Model and the Problem

In this paper, we address the SCATTERING problem within SILBOT, where particles act independently of each other, without explicit communication, in an asynchronous way, based only on local knowledge. SILBOT is a recent variant of the well-established geometric *Amoebot* model (see, e.g., [1, 9, 10, 14, 15]).

The Operating Environment. Particles operate on an infinite triangular grid embedded in the plane, where each node has 6 incident edges. Each node can contain at most one particle. There are N particles in the considered system and there might be nodes occupied by *obstacles*, i.e., immovable objects recognizable by the particles.

Particles and Configurations. Each particle is an automaton with two states, CONTRACTED or EXPANDED (they do not have any other form of persistent memory). In the former state, a particle occupies a single node of the grid while in the latter, the particle occupies one single node and one of the adjacent edges. Hence, a particle never occupies two or more nodes at once.

Each particle can sense its surroundings, i.e., if a particle occupies a node v, then it can see the neighbors' nodes of v (i.e., nodes at distance one). Specifically,

a particle can determine (i.e., sense) if a node is empty (i.e., not occupied by a particle nor an obstacle) or occupied by a CONTRACTED or an EXPANDED particle, or occupied by an obstacle, for any node in its direct neighborhood.

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

It is assumed that initially particles are all CONTRACTED.

Movement and States. Each particle can occupy only one node v at a time. In order to move to a neighboring node u, the particle expands on the edge between node v and node u. Thus, in the EXPANDED state, the particle occupies one node and one edge. Note that, node u may still be occupied by another particle. If the other particle leaves node u in the future, the EXPANDED particle will contract into node u during its next activation.
A particle commits itself into moving to node u by expanding in that direction, and at the next activation of the same particle, it is constrained to move to node u, if u is empty. A particle cannot revoke its expansion once committed.

Asynchrony and Rounds. The SILBOT model introduces a fine-grained notion of asynchrony with possible delays between observations and movements performed by the particles.[3] All operations performed by the particles are non-atomic: that is, there can be delays between the actions of sensing the surrounding, computing the next decision, executing the decision (i.e., change of state, movement, expansion, contraction).

There are no assumptions nor restrictions on the scheduling of these events; thus any possible execution of an actual physical system can be captured by the model, hence inducing many difficulties for proving the correctness of resolution algorithms (see, e.g. [4, 6]).

A *round* is any time window within which all particles have been activated and concluded their activation time at least once. When asynchrony is assumed, it is also required the well-established fairness property by which each particle is activated infinitely often in any execution of the particle system. Hence, the duration of a round is finite but unknown and may vary from time to time. Due to the asynchronous nature of the system, it may happen that a particle decides (or is forced, in case of contraction) at time t to take an action, and that this action will actually be executed at time $t' > t$, when other particles may have changed their state. The time required to accomplish an action is finite but unknown.

Orientation and Randomness. Particles are disoriented, we do not make any assumptions about the local coordinate system of a particle. It may even change in each activation of the particle. Furthermore, we aim at studying the case where particles take deterministic decisions. However, we will see in our simulation how randomness may play a central role.

It is worth noting that there are cases that cannot be deterministically resolved and are left to the power of the scheduler. For instance, if two CON-

[3] Somewhat similar to the so-called ASYNC model designed for theoretical models dealing with mobile and oblivious robots (see, e.g., [4–6, 16, 17]).

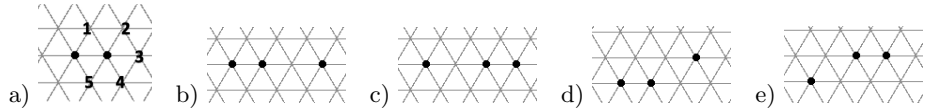

Fig. 1. a) two neighboring particles with the numbering of the unoccupied nodes neighboring the rightmost particle; b) a basic configuration where at least the middle particle must move in order to solve Scattering; c) a basic configuration similar to the one in the middle; d) and e) two similar configurations where at least the middle particle must move in order to solve Scattering.

tracted particles decide to expand on the same edge simultaneously, exactly one of them (arbitrarily chosen by the scheduler) succeeds. If two particles are expanded on two distinct edges incident to the same node w, exactly one of the particles (again, chosen arbitrarily by the scheduler) contracts to node w, while the other particle remains expanded.

The Scattering problem. We can now formally define the Scattering (or Distancing) problem, introducing two possible variants concerning the behavior of the particles with respect to obstacles.

Definition 1 (Scattering). *Given an initial configuration, possibly with obstacles, an algorithm solves the* Simple Distancing *problem if there exists a time t after which all particles remain* contracted *at distance at least two from any other particle. It solves the* Strong Distancing *problem if there exists a time t after which all particles remain* contracted *at distance at least two from both any other particle and any obstacle.*

Note that, the Strong Distancing variant of the problem can also be considered as a scenario where obstacles are actually crashed (i.e., non-moving) particles [11], recognizable from those working correctly.

3 Impossibility for Scattering within SILBOT

In this section, we show that within SILBOT, the Scattering problem cannot be solved. Hence, some more restrictions or particles' capabilities must be added.

Theorem 1. Scattering *is unsolvable within* SILBOT.

Proof. To prove the claim, it is sufficient to provide an example where the problem cannot be solved. First of all, it is worth noting that an isolated particle should not move. In fact, if an algorithm allows that movement, then the time t required in the Def. 1 would be never reached as any contracted particle can expand as soon as activated. Furthermore, any algorithm that solves the Scattering must necessarily provide a move for the particles of a (sub-)configuration made by just two neighboring particles like in Fig. 1.a, otherwise the problem would never be solved. Then, for that (sub-)configuration we can

basically specify two algorithms in order to guarantee to solve SCATTERING. In the first, say $\mathcal{A}'$, we assume a particle that is neighboring just another one, if activated, moves towards the opposite direction with respect to its neighbor. By referring to Fig. 1.a, the rightmost particle would move towards the position denoted by 3; in the second algorithm, say $\mathcal{A}''$, the same particle, if activated, expands and then moves towards one of the two symmetric positions 2 or 4, the adversary decides. The other possible expansions (towards positions 1 and 5 or even towards the neighboring particle) would always leave the particle neighboring the other one, hence we do not consider such cases as they do not resolve the SCATTERING. Let us consider Algorithm $\mathcal{A}'$ and the configuration in Fig. 1.b: the adversary can activate the particles in a (fair) way that brings the system in an infinite loop. In fact, it can activate the particle in the middle, that moves to the right, and the particle on the right, which doesn't move as it is isolated. Once the particle in the middle has moved (Fig. 1.c), the adversary can activate the particle on the left, which does not admit a neighbor, i.e., doesn't move, and the particle in the middle that would move back to its original position, hence creating the infinite loop.

When considering Algorithm $\mathcal{A}''$, similar arguments as above can be provided by starting from the configuration shown in Fig. 1.d (or Fig. 1.f). □

The above theorem confirms that in order to solve the SCATTERING within SILBOT, more restrictions to the environment or some more particles' capabilities are required. Enlarging the visibility range as in [7] doesn't seem to be effective.

An interesting case is to consider the asynchronous case with the assumption that a CONTRACTED particle is activated if it is (or becomes) neighboring to at least another particle. In practice, the adversary does not control anymore the activation of CONTRACTED particles which becomes an *event driven* (ED) process. The duration of an activation as well as the time required by a particle to accomplish an expansion or a contraction remains in the decision of the adversary as well as the activation of expanded particles. It turns out that this kind of schedule, that we call ED-ASYNC, is less general than the pure asynchronous one, but it is clearly more general than the synchronous case where all particles are always active. From a practical point of view, it is like the sensing abilities of the particles about their surrounding react to the stimuli given by neighboring particles by initiating the activation.

As we are going to show, this new assumption is enough to allow the resolution of the SCATTERING without obstacles. On the contrary, with obstacles, new difficulties must be faced and the problem turns out to be much harder.

Theorem 2. *Given a configuration with obstacles,* SCATTERING *is unsolvable within* SILBOT*, even when the synchronous schedule is assumed.*

Proof. To prove the claim, we provide an example where the problem cannot be solved for both its variants, even considering the synchronous scheduler which is a special case of both the asynchronous and the ED-ASYNC ones.

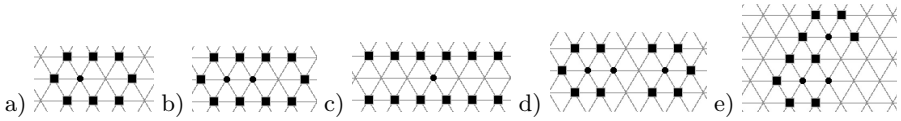

Fig. 2. Basic configurations with obstacles (black squares) used in the proof of Th. 2.

We first consider the Simple Distancing variant. In that context, it is worth noting that a particle in a situation like the one shown in Fig. 2.a should not move, as otherwise it would move forever back and forth, and the problem cannot be solved. Similarly, two particles "entrapped" in a (sub-)configuration like the one shown in Fig. 2.b should not move concurrently, as otherwise they would move forever back and forth, and again the problem cannot be solved.

Considering instead Fig. 2.d, the particle in the middle should move in order to solve Simple Distancing, whereas its neighboring particle as well as the other one cannot move because of the above arguments. If the allowed move is towards the right (position 3 according to Fig. 1), then a similar configuration would be achieved, from which the same particle would move back to its original position, i.e., an infinite loop occurs. Hence, the move should be towards one of the other two possible directions (positions 2 and 4 according to Fig. 1). However, starting from the configuration shown in Fig. 2.e, similar arguments as above can be provided, showing that the system can enter in an infinite loop.

Concerning Strong Distancing, it is enough to consider the configuration of Fig. 2.c. In fact, the particle must move in order to exit the tunnel but the adversary can make it move back and forth since the particle is disoriented. □

As seen above, synchrony doesn't help in the resolution of Scattering when obstacles occur. The proof may suggest that a crucial point is about the orientation of the particles. However, when obstacles are considered, we may think about particles entrapped inside a sort of labyrinth constituted by the obstacles, where there is just one exit. However, even considering particles endowed with chirality, i.e., a common clockwise direction, still there might be unsolvable configurations. It is sufficient to consider the case where a particle needs to traverse a *tunnel*, i.e., as in Fig. 2.c, a corridor of one node width. In fact, in such occurrences, a particle cannot deduce whether it was coming from the left or from the right, since the same node may assume the role of predecessor or successor with respect to the direction of exploration of the labyrinth. This is particularly crucial for the Strong Distancing variant, as even an 'isolated' particle inside a tunnel should move in order to find a position far apart from the obstacles.

In Section 5, we conduct some simulations in order to understand how much randomness may affect the solvability of Scattering with obstacles.

4 Scattering without obstacles

In this section, we define a very simple algorithm to solve the Scattering problem when no obstacle is present. According to Th. 1, we need to add some

capabilities to the particles or restrict the environment in order to allow the resolution of the problem. As anticipated in the previous section, we consider the ED-ASYNC schedule where a CONTRACTED particle is activated as soon as it is (or becomes) neighboring to at least another particle. The duration of an activation as well as the time required by a particle to accomplish an expansion or a contraction remains in the scope of the adversary as well as the activation of EXPANDED particles.

– Algorithm $\mathcal{A}$: Given a CONTRACTED particle p, let I be the maximal interval of consecutive neighboring nodes of p where no other particles lie. If $2 < |I| < 6$, then p expands towards the most central node in I – the rightmost one in case of ties.

We remind that particles do not share any common orientation, hence the term "rightmost" used in the algorithm can be interpreted differently by each particle according to its own local coordinate system. Moreover, the algorithm does not refer to EXPANDED particles since an EXPANDED particle can only reach its destination as soon as it is activated and the node towards which it is expanded gets empty, i.e., we have no control on it.

4.1 Aligned Particles

In this section, we consider the case where N particles are all aligned, not necessarily all adjacent within an N-nodes segment. Let S be the smallest segment containing all the particles. It is worth noting how this case is well-related to the Ants on a Stick puzzle according to the specified algorithm, see [24]. In fact, the interaction of the particles reminds the bounce event occurring among ants moving in opposite directions. The main difference is that particles only move if close to each other, i.e., they can reach the end of the stick (segment S) only if stimulated/pushed by other particles. Actually, like for ants, the relative order of the particles never changes as overtaking is not possible.

Lemma 1. *After at most $N - 1$ expansions of the leftmost and the rightmost particles of S, such particles never expand again.*

Proof. Let us assume p is the leftmost particle in S. First note that, if p expands, according to our algorithm, it can only expand leftward. Also, according to the assumed scheduler, when p is CONTRACTED, it expands leftward as soon as its (rightward) neighboring node along S is occupied. After p expands and then becomes CONTRACTED, eventually, its neighboring node (i.e., the one previously occupied by p itself) becomes empty, unless the particle p', the one that forced the expansion of p, returns to be neighboring to p because of another expansion forced by another particle, the third one on S, counting from the left bound. Thus, before p can expand again, i.e., enlarging S again, p' must reach the neighboring node of p by means of an expansion generated by the particle to the right of p'.

By iterating the above argument, since each new expansion of p involves one new particle, we can conclude that p can expand at most $N-1$ times. A similar argument can be applied to the rightmost particle. □

The above lemma can be exploited inductively on the number of particles for proving the next result.

Theorem 3. *When N particles are all aligned, Algorithm $\mathcal{A}$ solves the* SCATTERING *problem.*

Proof. We prove the claim by induction on the number of particles. For $N \leq 2$ particles the claim trivially holds. Assume the claim for $N-1$ particles.

By Lemma 1, Algorithm $\mathcal{A}$ allows the outermost particles p and p' to expand a finite number of times, upper bounded by $N-1$. Consider the time after which p and p' cannot expand anymore, i.e., according to the assumed scheduler, no particles become neighbors of p and p' anymore. We can restrict our attention to the remaining $N-2$ particles obtained by excluding p and p'. By the inductive hypothesis, Algorithm $\mathcal{A}$, applied on such $N-2$ particles, resolves the SCATTERING. Moreover, by hypothesis, their final positioning cannot induce further expansions for p and p', i.e., p and p' admit no neighboring particles. Hence, the SCATTERING is solved for all the N particles. □

4.2 General configurations

We consider now the general case where the initial configuration is constituted by any placement of N CONTRACTED particles over any N nodes of the triangular grid. By similar arguments than those applied in the previous section, we prove that Algorithm $\mathcal{A}$ solves the SCATTERING problem also in the general setting.

Instead of the segment S specified in the previous section, given the initial configuration, we consider the smallest regular hexagon H_i centered on a node of the grid and containing all the particles. Let c be the center of H_i; the index i represents the distance in terms of hops between c and the perimeter of H_i (refer also to the example depicted in Fig. 3.a). Note that H_i is not necessarily unique; however, we just need to fix one for our analysis purposes. Hence, particles are not required to be aware about H_i nor c.

As a first observation, a particle on the perimeter of H_i cannot expand towards the inner part of H_i according to Algorithm $\mathcal{A}$. This can be easily verified by means of a case analysis. As for notation, by H_{i+1} we denote the regular hexagon centered in c with one level more with respect to H_i; similarly, by H_{i-1} we denote the regular hexagon centered in c with one level less with respect to H_i.

Lemma 2. *Let H_j be the smallest hexagon centered in c, with $j \geq i$, that contains all particles. Particles on H_j, within finite time, either move to H_{j+1} or they are all isolated (admitting no neighboring particles), eventually.*

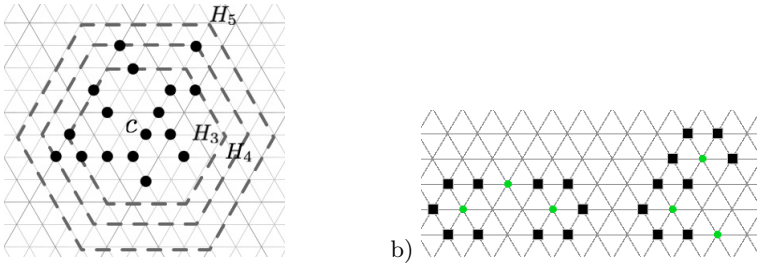

Fig. 3. a) a configuration with H_4 as the smallest enclosing hexagon; b) two possible solutions for SIMPLE DISTANCING obtainable from the configurations of Fig.s 2.d and 2.e. Green circles represent particles without neighboring other particles.

Proof. Apart from isolated particles, the only case in which a particle p on H_j cannot move according to Algorithm $\mathcal{A}$ is when it admits two neighboring particles p' and p'' residing on H_j (possibly with other neighboring particles on H_{j-1}) and p does not occupy a corner of H_j. However, such a situation cannot occur for all the non-isolated particles on H_j since p' and p'' must be in the same situation of p, creating a chain of particles on H_j. In order to conclude the proof, it is sufficient to observe that if a corner of H_j is occupied, it admits at least three consecutive neighboring nodes empty, i.e., according to Algorithm $\mathcal{A}$, the particle on the corner either is isolated or it will move, eventually. □

By the above lemma and similarly to Lemma 1, we prove the next result.

Lemma 3. *A particle on H_i can increase its distance from c at most $N-1$ times.*

Proof. A particle p on H_i, in order to move from H_i to H_{i+1}, must be neighboring by at least another particle p' residing either on H_{i-1} or H_i. Successively, once particle p becomes CONTRACTED again, it can move from H_{i+1} to H_{i+2} if:

- either it becomes neighboring to a particle on H_i, initially located on a node different from the one previously occupied by p' (hence occupied by a third particle p'');
- or it becomes neighboring to a particle that has also reached H_{i+1}, pushed itself by at least another particle on H_i (since H_{i+1} has no particles on itself at the beginning).

As done for the case of aligned particles of Lemma 1, by iterating this argument, the claim holds. □

Furthermore, by exploiting Lemma 3, we prove the next result inductively on the number of particles.

Theorem 4. *Algorithm $\mathcal{A}$ is correct and terminates.*

Proof. We prove the claim by induction on the number of particles. For $N \leq 2$ particles the claim is easy to see. Assume the claim for $N-1$ particles, we prove it for N.

By Lemma 3, Algorithm $\mathcal{A}$ allows the outermost particles to move away from H_i a finite number of times, upper bounded by $N-1$. At this time, all particles on the outermost regular hexagon centered in c, say H_j, cannot move any more; hence, by Lemma 2, all particles on H_j are isolated. Consider the time after which particles on H_j do not expand anymore, nor new particles reach this hexagon. Let us consider now the instance given by all the particles that are inside H_j and not on it. By inductive hypothesis, since these particles are less than N, Algorithm $\mathcal{A}$ resolves the SCATTERING. Moreover, by hypothesis, the final positioning of the involved particles cannot induce further expansions for the particles on H_j, i.e., particles on such a hexagon are all isolated. Hence, the SCATTERING is solved for all the N particles. □

5 Approaching SCATTERING with obstacles

In Section 3, we have proven the SCATTERING problem is unsolvable within SILBOT when obstacles are present in the configuration, even in the synchronous setting. Here we aim to approach the problem in the asynchronous setting (not even the ED-ASYNC) but relaxing the power of the adversary. We consider the case where the fairness of the scheduler is randomized, and not addressed to detect the worst case scenario. In practice, although in general the configurations shown in Fig.s 2.d and 2.e remain unsolvable, if the scheduler does not play the adversarial role but "fairly" allows the particles to reach symmetric destinations when ties occur, then the situations can be managed differently. Note that, this kind of scheduler is not equivalent to let particles approach random walks. Algorithms are still deterministic but "sometimes" the output, combined with the scheduler, mimics a random choice. In fact, when there are equivalent neighbors for a robot where to move, it selects the destination according to its own local coordinate system that can change from one activation to another. In other words, the "rightmost" node chosen by a particle when applying our deterministic algorithms may change from one activation to another because the local coordinate system changes. To this respect, we have developed a computer simulation framework for approaching both the SIMPLE DISTANCING and the STRONG DISTANCING variants, see [2, 21].

First of all, it is worth noting that when obstacles are considered, an algorithm similar to that provided in Section 4 cannot succeed. It is sufficient to consider the configuration of Fig. 2.d where the middle particle may move back and forth forever. Hence we need to define a new algorithm.

5.1 The SIMPLE DISTANCING problem

Given a particle p, we remind the reader that I is defined as the maximal interval of consecutive neighboring nodes of p where no other particles lie. Note that,

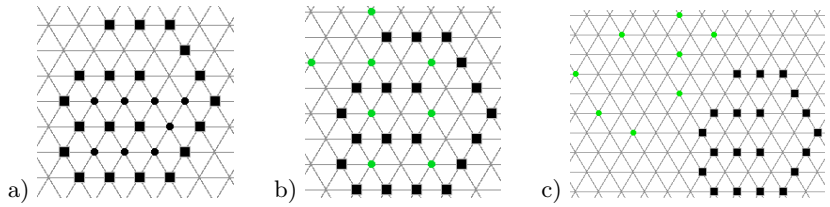

Fig. 4. a) a very constrained configuration; b) the corresponding solution obtained in a run of the algorithm for Simple Distancing; c) the corresponding solution obtained in a run of the algorithm for Strong Distancing.

in the computation of I, obstacles are treated as empty nodes. Our algorithm works as follows. In general, we fix a direction towards which the particle should expand. Ties are resolved by the local perception of a particle, giving priority to the rightmost direction. We remind that particles do not agree on any direction, and moreover a particle may have a different knowledge/perception with respect to different activations. Once chosen a direction, if an obstacle is present towards that direction, then a neighboring direction is tested until detecting an empty node, if any. Otherwise the particle does not expand. The directions provided by the algorithm are the following:

- If $|I| = 1$, then p tries to expand towards the only node in I;
- If $|I| = 2$, then p tries to expand towards the rightmost node in I;
- If $|I| = 3$ or $|I| = 5$, then p tries to expand towards the rightmost node among the two neighbors of the central node in I;
- If $|I| = 4$, then p tries to expand towards the rightmost node among the two central ones in I.

Fig. 3.b shows the obtained solutions when starting from the configurations of Fig.s 2.d and 2.e. We have run our simulator on various instances where the resolution of the Simple Distancing is rather constrained due to the presence of obstacles. Keeping in mind the commitment of expanded particles, one may expect deadlock situations when the density of the particles is high whereas the moving space is reduced, as in Fig.s 4.a and 5.a. Our runs repeated several times always obtained a successful behavior. Fig.s 4.b and 5.b show two possible solutions obtained from the configurations shown in Fig.s 4.a and 5.a, respectively, even though the solution for Fig. 5.a may require a large number of rounds (we experienced about 2200 rounds on average) to converge. In fact, there exists exactly one solution according to the number of particles and the shape of unoccupied nodes.

5.2 The Strong Distancing problem

In this scenario, the goal of the particles is to be distanced also from the obstacles (possibly crashed particles), see Def. 1.

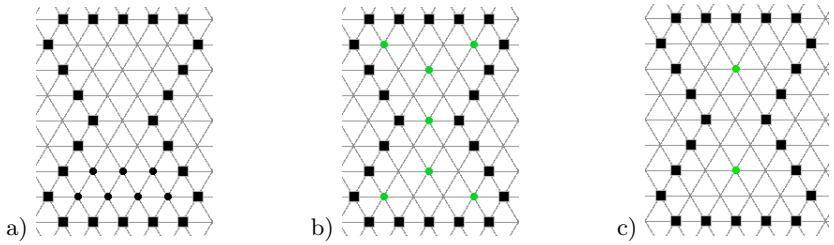

Fig. 5. a) a very constrained configuration; b) the corresponding solution obtained in a run of the algorithm for Simple Distancing; c) a solution obtained for Strong Distancing with the maximum number of particles allowing to solve the problem within the area confined by the obstacles.

The basic algorithm we tested in this case is similar to the one proposed in the previous section, with the additional move required for a particle p when $|I| = 6$ with at least an obstacle neighboring p, i.e., p is not isolated from obstacles:

- if $|I| = 6$ and p is neighboring to at least an obstacle, then p tries to expand towards the rightmost neighboring node.

We remind that if an obstacle is present in the direction chosen by the algorithm, then a neighboring direction is tested until detecting an empty node, if any, otherwise the particle doesn't expand. Clearly, in order to solve Strong Distancing, there cannot exist, in the initial configuration, a particle surrounded by 6 obstacles or a situation like in Fig. 2.a where an area enclosed by obstacles does not allow the distancing of the particles.

The proposed algorithm is clearly still able to successfully solve the specific configurations shown in Fig.s 2.d and 2.e. Also, it succeeds in configurations similar to that shown in Fig. 5, assuming there is enough space inside the area delimited by the obstacles to place all the particles distanced both from each other and from the obstacles. In the specific case, only two particles could be placed inside the area delimited by the obstacles, see, e.g., the obtained solution in Fig. 5.c. However, the algorithm may find quite some difficulties with the configuration of Fig. 4.a even though we always experienced successful runs requiring about 2500 rounds on average. A corresponding solution is shown in Fig. 4.c. The ability of the particles to all exit the labyrinth formed by the obstacles mainly depends on the direction that is close to be randomly chosen to make the particles expand, especially when their only neighbors are obstacles (i.e., $|I| = 6$). Here, some randomization arguments might be required to understand the behaviour of the particles.

When considering, instead, configurations where the placement of the objects does not constrain much the movement of the particles, the algorithm results to be quite effective. We executed over 1000 runs of the simulator as follows: at each run, about 60 particles and 40 obstacles were randomly placed in an initial area of diameter of 18 edges. The particles were able to complete the task with an average number of about 240 rounds.

6 Conclusion

We have considered the SCATTERING problem with programmable matter within the very weak SILBOT model. We have shown resolution algorithms and simulations for the different variants of the problem. Several questions are still open, and leave space for further investigations: for instance, to prove whether our algorithms for SCATTERING with obstacles are correct, maybe by introducing some probability arguments. It would be interesting to detect the minimal set of assumptions and/or constraints on the obstacle placement under which SCATTERING with obstacles can be solved, in both its variants. Finally, it is worth investigating the general case of faulty particles, where it is not possible to distinguish between crashed and correctly working particles.

References

1. Bazzi, R.A., Briones, J.L.: Deterministic leader election in self-organizing particle systems. In: Proceedings of the 21st International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS). vol. 11914. Springer (2019)
2. Bonini, S.: Programmable matter simulator. `https://github.com/samul-1/unipi_programmable_matter` (2022)
3. Bramas, Q., Tixeuil, S.: The random bit complexity of mobile robots scattering. Int. J. Found. Comput. Sci. **28**(2), 111–134 (2017)
4. Cicerone, S., Di Stefano, G., Navarra, A.: Asynchronous arbitrary pattern formation: the effects of a rigorous approach. Distributed Computing **32**(2), 91–132 (2019)
5. Cicerone, S., Di Stefano, G., Navarra, A.: Solving the pattern formation by mobile robots with chirality. IEEE Access **9**, 88177–88204 (2021). https://doi.org/10.1109/ACCESS.2021.3089081
6. Cicerone, S., Di Stefano, G., Navarra, A.: A structured methodology for designing distributed algorithms for mobile entities. Information Sciences **574**, 111–132 (2021). https://doi.org/10.1016/j.ins.2021.05.043
7. D'Angelo, G., D'Emidio, M., Das, S., Navarra, A., Prencipe, G.: Asynchronous silent programmable matter achieves leader election and compaction. IEEE Access **8**, 207619–207634 (2020)
8. Daymude, J.J., Derakhshandeh, Z., Gmyr, R., Porter, A., Richa, A.W., Scheideler, C., Strothmann, T.: On the runtime of universal coating for programmable matter. Natural Computing **17**(1), 81–96 (2018)
9. Daymude, J.J., Gmyr, R., Hinnenthal, K., Kostitsyna, I., Scheideler, C., Richa, A.W.: Convex hull formation for programmable matter. ICDCN 2020, ACM (2020)
10. Derakhshandeh, Z., Gmyr, R., Strothmann, T., Bazzi, R.A., Richa, A.W., Scheideler, C.: Leader election and shape formation with self-organizing programmable matter. In: Proceedings of 21st International Conf. on DNA Computing and Molecular Programming (DNA). vol. 9211, pp. 117–132. Springer (2015)
11. Di Luna, G.A., Flocchini, P., Prencipe, G., Santoro, N., Viglietta, G.: Line recovery by programmable particles. In: Proceedings of the 19th International Conf. on Distributed Computing and Networking, (ICDCN). pp. 4:1–4:10. ACM (2018)
12. Di Luna, G.A., Flocchini, P., Santoro, N., Viglietta, G., Yamauchi, Y.: Shape formation by programmable particles. Distributed Comput. **33**(1), 69–101 (2020). https://doi.org/10.1007/s00446-019-00350-6

13. Dolev, S., Frenkel, S., Rosenblit, M., Narayanan, R., Venkateswarlu, K.M.: In-vivo energy harvesting nano robots. In: 2016 IEEE International Conference on the Science of Electrical Engineering (ICSEE) (2016). https://doi.org/doi:10.1109/icsee.2016.7806107
14. Dufoulon, F., Kutten, S., Moses Jr., W.K.: Efficient deterministic leader election for programmable matter. In: Miller, A., Censor-Hillel, K., Korhonen, J.H. (eds.) PODC: ACM Symposium on Principles of Distributed Computing. pp. 103–113. ACM (2021)
15. Emek, Y., Kutten, S., Lavi, R., Moses Jr., W.K.: Deterministic Leader Election in Programmable Matter. In: Baier, C., Chatzigiannakis, I., Flocchini, P., Leonardi, S. (eds.) 46th International Colloquium on Automata, Languages, and Programming (ICALP 2019). Leibniz International Proceedings in Informatics (LIPIcs), vol. 132, pp. 140:1–140:14. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany (2019). https://doi.org/10.4230/LIPIcs.ICALP.2019.140
16. Flocchini, P., Prencipe, G., Santoro, N.: Moving and computing models: Robots. In: Distributed Computing by Mobile Entities, vol. 11340 LNCS, pp. 3–14. Springer (2019)
17. Flocchini, P., Prencipe, G., Santoro, N., Viglietta, G.: Distributed computing by mobile robots: uniform circle formation. Distributed Computing **30**(6), 413–457 (2017)
18. Izumi, T., Kaino, D., Potop-Butucaru, M.G., Tixeuil, S.: On time complexity for connectivity-preserving scattering of mobile robots. Theor. Comput. Sci. **738**, 42–52 (2018)
19. Miyashita, S., S., S.G., Li, Rus, D.: Robotic metamorphosis by origami exoskeletons. Science Robotics **2**(10) (2017)
20. Patitz, M.J.: An introduction to tile-based self-assembly and a survey of recent results. Nat Computing **13**, 195—-224 (2013)
21. Tracolli, M.: Programmable matter simulator. `https://github.com/MircoT/programmable-matter-simulator` (2022)
22. 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. pp. 550–558. AAMAS '18, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC (2018)
23. Walter, J., Welch, J., Amato, N.: Distributed reconfiguration of metamorphic robot chains. Distributed Computing **17** (2004). https://doi.org/doi:10.1007/s00446-003-0103-y
24. Winkler, P.: The adventures of ant alice. In: A Lifetime of Puzzles, pp. 177–185. CRC Press (2008)