Theoretical Computer Science 996 (2024) 114510

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Molecular pattern formation on grids in the MOBLOT model ☆,☆☆,☆☆☆

Serafino Cicerone [a], Alessia Di Fonso [a], Gabriele Di Stefano [a], Alfredo Navarra [b,*]

[a] Department of Information Engineering, Computer Science and Mathematics - University of L'Aquila, Via Vetoio, L'Aquila, I–67100, Italy
[b] Department of Mathematics and Computer Science - University of Perugia, Via Vanvitelli 1, Perugia, I-06123, Italy

ARTICLE INFO

Communicated by S. Dolev

Keywords:
Distributed algorithms
Mobile robots
Pattern formation
Oblivious robots
Grid graph

ABSTRACT

In the theoretical studies on distributed algorithms for swarm robotics, the complexity and capabilities of the robots are usually reduced to their minimum. Recently, the MOBLOT model has been introduced in order to deal with robots considered silent, anonymous, and oblivious but capable to aggregate into more complex structures, called *molecules*. We study the case where robots move along a graph based on a square lattice and we formally define the *Molecular Pattern Formation* (MPF) problem, where a specific configuration of robots assembled into molecules must be reached. As a preliminary general result, we provide a necessary condition for its solvability. Then, we actually show that dealing with molecules can resolve in some cases the symmetry breaking issue on grids where otherwise robots cannot. Finally, we introduce an interesting case study, representative of the MPF problem, in which the molecules can be formed by the set of the seven tetrominoes (aka Tetris blocks). We provide a complete characterization of this specific problem, providing a distributed algorithm able to form a molecular pattern whenever the necessary condition for the solvability of MPF is verified.

1. Introduction

Robotics is an active area of research that includes many computer science and engineering disciplines. Dealing with robotics concerns the design, the construction, the operation, and the use of robots. In particular, two main research areas have been deeply investigated in robotics: *modular robotics* (e.g., see [2]) and *swarm robotics* (e.g., see [3,4]). Swarm robotics differ from modular robotics as single robots in the system do not need to maintain the connection with each other at all times, but they are usually mobile units with full autonomy (e.g., Kilobot [5]). In such a context, the interaction among robots in some specific form should lead to a desired collective behavior. This approach emerged in the field of artificial swarm intelligence, as well as the biological studies of insects, ants and other fields in nature, where swarm behavior occurs.

Researchers in the field of swarm robotics mainly follow a theoretical approach that considers robot systems in the abstract, where the complexity and capabilities of the robots induced by the underlying model are often reduced to their minimum. One of the

☆ This article belongs to Section A: Algorithms, automata, complexity and games, Edited by Paul Spirakis.

☆☆ Preliminary results appeared in the Proceedings of the 18*th* International Symposium on Algorithmics of Wireless Networks (ALGOSENSORS) 2022 [1].

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

* Corresponding author.
E-mail addresses: serafino.cicerone@univaq.it (S. Cicerone), alessia.difonso@univaq.it (A. Di Fonso), gabriele.distefano@univaq.it (G. Di Stefano), alfredo.navarra@unipg.it (A. Navarra).

https://doi.org/10.1016/j.tcs.2024.114510
Received 6 February 2023; Received in revised form 4 January 2024; Accepted 14 March 2024

Available online 20 March 2024

main issues faced when dealing with such models is that they help, in general, to rigorously analyze the designed algorithms, hence providing new theoretical insights that subsequently also extend the practical aspect of the studied systems. Representative models in this context are the Amoebot model [6–8], the more recent models Silbot [9–13] and Pairbot [14], and the well-investigated OBLOT (see, e.g., [15]).

The OBLOT model deals with the case of distributed robots moving in some environment, and can be considered as a sort of framework within which many different settings can be manipulated, each implied by specific choices among a range of possibilities, with respect to fundamental components like time synchronization as well as other important elements, such as memory, orientation, mobility and communication. Such settings are often maintained at minimum, thus defining very weak computational entities: robots are *oblivious* (no memory about past activities), *identical* (indistinguishable from their appearance), *anonymous* (no distinct identities that can be used during the computation), *autonomous* (they operate without a central control or external supervision), *homogeneous* (they all execute the same algorithm), *silent* (they have no means of direct communication with each other - everything is computed on the base of the relative positions of the robots only), and *disoriented* (each robot operates in its own local coordinate system). When a robot is activated, it enters in a so-called `Look–Compute–Move` cycle: it acquires a snapshot of the current global configuration (`Look`) as the disposal of the robots with respect to its own coordinate system. Successively, it decides whether to move toward a specific target or not (`Compute`), and in the positive case it moves (`Move`). In general, robot's capabilities are maintained as weak as possible so as to understand what is the limit for the feasibility of the problems. Moreover, the less assumptions are made, the more a resolution algorithm is robust with respect to possible disruptions.

In this work, we consider MOBLOT [15], a recently introduced theoretical model concerning swarm robotics that extends OBLOT to address a larger spectrum of problems. MOBLOT stands for *Molecular OBLivious robOTs*: like atoms combine themselves to form molecules, in MOBLOT simple robots can bond with each other in order to create possibly bigger computational units with more intrinsic capabilities with respect to robots (called *molecules* also in the model). A molecule is specified by a pattern. When robots move so as to be correctly positioned according to the pattern they firmly bond to make a molecular robot; like in nature, molecules can further bond to create more complex structures (e.g., the matter), the MOBLOT version of molecules can exploit their own capabilities to accomplish new tasks to form any shape defined according to some compositional properties or specific patterns.

Our results. MOBLOT has been initially defined for entities (robots and molecules) moving in a continuous environment (the Euclidean plane). Motivated by the observation that many models for swarm robotics assume robots moving in a graph (e.g., the Amoebot model uses a graph defined by the triangular lattice), as a first contribution here we extend MOBLOT for working also in a discrete environment based on a graph defined by the square lattice. Another observation that guides this work concerns the main problem faced in the OBLOT model, that is *Pattern Formation* [16–24] (see also the recent survey in [25] and references therein): given a team of robots and a geometric pattern in terms of points in the plain with respect to an ideal coordinate system (not known to the robots), the goal is to design a distributed algorithm that works for each robot to guide it so that eventually all robots together form the pattern, if possible. Based on this observation, here we extend this problem in the MOBLOT model by defining the *Molecular Pattern Formation* (MPF) problem: given a team of robots, the definition of a set of molecules (ideally small patterns composed by a few of robots), and the matter to be formed (defined according to some adjacency properties among molecules or by providing a specific pattern made of molecules), the aim is to provide an algorithm that works for each single robot and for each composed molecule, in a distributed way, to guide them so that eventually the matter is formed, if possible. As usual, both the molecule and the matter are defined with respect to an ideal coordinate system (not known to the robots nor to the molecules).

Determining which patterns are formable under what conditions has been the subject of extensive studies in a variety of settings. Hence, as a general result, we provide a necessary condition for the solvability of MPF which relies on two factors. The first is somehow inherited from the continuous case, that is the symmetricity. Informally, the symmetricity of a configuration measures the amount of symmetries of the robots' disposal. The second is instead specific of the discrete environment and concerns the type of center of the smallest enclosing box of the pattern. As we deal with regular square grids, such a center might be a vertex, the middle point of an edge, or the center of a square of the grid. We then show that dealing with molecules can resolve in some cases the symmetry breaking issue on grids where OBLOT cannot. Finally, we introduce an interesting case study, representative of the MPF problem, in which the set of potentially formable molecules is the set of the seven tetrominoes (aka Tetris blocks) [26]. This case study is a specific variant of MPF and is called Tetris-like MPF (TL-MPF). For this problem, we provide a complete characterization, that is we provide a distributed algorithm for a set of synchronous robots that is able to form a molecular pattern whenever the necessary condition for the solvability of MPF is verified.

Outline. Since the MOBLOT model is an extension of OBLOT, in the next section we first recall the latter model and then we provide the MOBLOT model on square lattice. The section is concluded with the formal definition of the MPF problem. Section 3 contains the necessary condition for the feasibility of MPF. In Section 4, we introduce the Tetris-like MPF problem, and in Section 5 we formalize $\mathcal{A}_{\mathsf{TL}}$, the resolving algorithm for TL-MPF. In Section 6, we give an extended example to show the effectiveness of $\mathcal{A}_{\mathsf{TL}}$, whereas in Section 7 we formally provide its correctness. Section 8 concludes the paper by highlighting some final remarks.

2. Molecular oblivious robots on grids

In this section, we present the MOBLOT model for robots moving on a square grid. This is provided by revising most of the properties characterizing the MOBLOT model introduced in [27] for robots moving on the Euclidean plane, which in turn was an extension of the OBLOT model for the same environment.

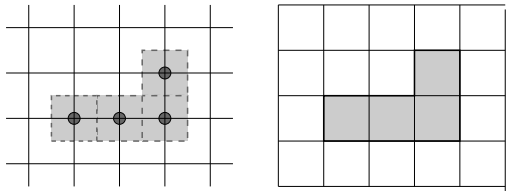

Fig. 1. A molecule formed by four adjacent robots and the dual grid, where only the extent of the molecule is shown.

An OBLOT system is composed by a set $R = \{r_1, r_2, \ldots, r_n\}$ of robots, that live and operate in graphs. Robots are viewed as points (they are **dimensionless**), and more than one robot can occupy the same vertex at the same time, i.e., a **multiplicity** can occur. They have the following basic properties: **typed** (each robot has a type that can be modeled as a color (e.g., see [28]), and when just one color is available we say that robots are **identical**), **anonymous** (they do not have distinct ids), **autonomous** (they operate without a central control or external supervision), **homogeneous** (they all execute the same algorithm), **silent** (they have no means of direct communication of information to other robots), and **disoriented** (each robot has its own local coordinate system - LCS) but they share a common handedness, i.e., *chirality* is assumed. A robot is capable of observing the positions (expressed in its LCS) of all the robots. We consider synchronous robots whose behavior follows the sequential phases that form a **computational cycle**:

- `Wait`. The robot is idle. A robot cannot stay indefinitely idle.
- `Look`. The robot obtains a snapshot of the positions of all the other robots expressed in its own LCS.
- `Compute`. The robot performs a local computation according to a deterministic algorithm $\mathcal{A}$ (i.e., the robot executes $\mathcal{A}$), which is the same for all robots, and the output is a vertex among its neighbors or the one where it resides.
- `Move`. The robot performs a *nil* movement if the destination is its current location otherwise it instantaneously moves on the computed neighbor.

`Look-Compute-Move` computational cycles (LCM cycles, for short) can be performed by robots according to different schedulers. In the **asynchronous** scheduler (ASYNC) the robots are activated independently, and the duration of each phase is finite but unpredictable (the activation of each robot can be thought as decided by the adversary). As a result, robots do not have a common notion of time. Moreover, according to the definition of the `Look` phase, a robot does not perceive whether other robots are intended to move or not. Hence, robots may move based on outdated perceptions. In fact, due to asynchrony, by the time a robot takes a snapshot of the configuration, this might have drastically changed once the robot starts moving. In the **fully-synchronous** scheduler (FSYNC) all robots are always active, and the activation phase can be logically divided into global rounds: for all $i \geq 1$, all robots start the i-th LCM cycle simultaneously and synchronously execute each phase. The **semi-synchronous** (SSYNC) scheduler coincides with the FSYNC one, with the only difference that some robots may not start the i-th LCM cycle for some i (some of the robots might be in the `Wait` state), but all of those who have started the i-th cycle synchronously execute each phase.

Robots are **oblivious**, that is they have no memory of past events. This implies that the `Compute` phase is based only on what determined in their current cycle (in particular, from the snapshot acquired in the current `Look` phase). A data structure containing all the information elaborated from the current snapshot represents what later is called the **view** of a robot. Since each robot refers to its own LCS, the view cannot exploit absolute orienteering but it is based on relative positions of robots. Hence, if symmetries occur, then symmetric robots have the same view. In turn, (i) the algorithm cannot distinguish between symmetric robots (even when placed in different positions), and (ii) symmetric robots may compute the same movements. As chirality is assumed, and we are considering a regular square grid as a field of movement, the only possible symmetries are rotations of 90 or 180 degrees.

MOBLOT extends the above system according to the *matter formation paradigm*: atoms (the smallest units of the matter) first combine with other atoms to form molecules (special kind of atom compounds) and then the molecules combine with each other to form some kind of matter, like for instance a crystal. Accordingly, in a MOBLOT system the smallest units correspond to the robots of the OBLOT model: each robot can be thought as an atom. As in nature there exist different types of atoms, in OBLOT we use colors to specify the type of atom the robot represents.

In MOBLOT, the first algorithmic task for robots is to form **molecules**. A molecule μ is specified by a **fixed pattern** (i.e., a specific reciprocal disposal of a specific subset of robots) defined with respect to the regular square grid. Once a molecule is formed, it is assumed to have an extent $B(\mu)$ given by the closed set given by the union of the squares of the same dimensions of the squares of the grid, each one centered on one of the robots composing the molecule. The center of the molecule corresponds to the center of the smallest rectangle enclosing all robots forming that molecule. Fig. 1 shows a molecule as formed by four robots and also the same molecule represented in the **dual grid**, where - for the sake of simplicity - only the extent of the molecule is considered.

To make the model fair and as general/weak as possible, we make some assumptions about the initial disposal of robots and the formation of molecules, as motivated by the following analysis. We assume that initially, each robot has no neighbors (it is similar to the typical assumption made in the OBLOT model where the robots are initially positioned on distinct vertices). This guarantees that there are no molecules already formed in the initial configuration and that their formation is completely up to the algorithm. During the execution of an algorithm, robots might become neighbors and potentially their disposal might seem to make them participate in two different molecules. However, the intersection between two molecules (including their boundaries) must be empty while at

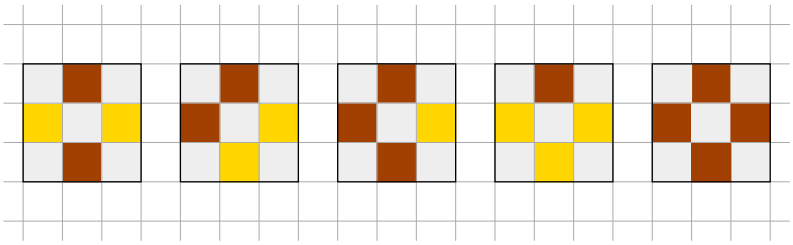

Fig. 2. A set of molecules formed by nine robots each (example taken from the notion of tiles described in [29]). There are three types of robots which are visualized by three different colors (brown, gold, and light gray). (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

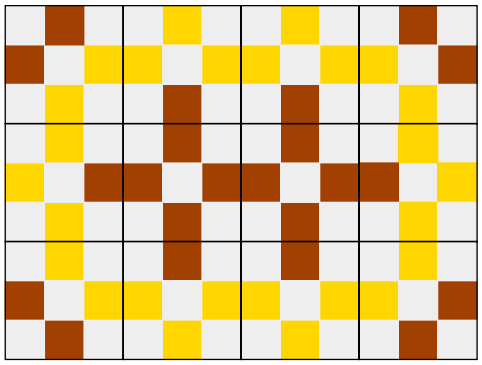

Fig. 3. An example of matter built by using molecules from Fig. 2 and defined according to adjacency properties.

least one of them is going to be formed, otherwise, the generation of that molecule is prevented. These assumptions can be modeled according to the following constraints:

C_1: in each initial configuration, each pair of robots is at distance not less than 2, where the distance is the number of edges composing the shortest path connecting them (this ensures that there are no molecules already formed in the initial configuration);

C_2: a molecule μ is formed as soon as the robots needed to build it are suitably placed according to the pattern that defines μ, and in $B(\mu)$ there must be no more robots than necessary (this ensures that the right number of robots are involved in the formation of a molecule);

C_3: for each μ just formed and for any other μ' already formed or that could be formed at the same time of μ, $B(\mu) \cap B(\mu') = \emptyset$ (this avoids ambiguities during the formation of molecules)

In MOBLOT, as soon as a molecule μ is formed, each robot forming μ is no longer an independent entity but it stops executing its algorithm as a single robot and acts as a part of the molecule. This means that once a molecule is formed it constitutes a **new computational entity** with a physical dimension, i.e., it is intended as solid. The basic properties of such new entities can still be modeled as in OBLOT systems (and its variants), with the main exception that a molecule not only can move by following an edge of the grid but it also may rotate of 90 degrees with respect to one of the vertices occupied by the robots composing it. Being solid, any other entity in the system (robot or molecule) can touch the external surface of $B(\mu)$ but cannot penetrate inside. It is worth to remark that in MOBLOT, a robot r performing the `Look` phase is able to detect not only all the other robots but also any formed molecule μ. Note that, r perceives both the molecule and the robots that compose it. We denote by *Mol* the set of molecules detected at a given time by a robot r.

By $\mathcal{M} = \{\mu_1, \mu_2, \ldots, \mu_m\}$ we denote the set containing all different molecules that can be potentially formed. For instance, in Fig. 2, five molecules are shown as displayed in the dual grid. Each molecule is formed by nine robots, and there are three different types of robots (visualized by means of three different colors). The five different molecules shown in Fig. 2 may constitute the set $\mathcal{M}$ given in input to the algorithm, and the algorithm is responsible to assemble all the molecules so that a more complex structure (i.e., the *matter*) is formed. The matter is defined either according to some adjacency properties of the molecules or by providing a fixed pattern made of molecules (cf., Fig. 3). By $\mathcal{F}$ we denote the set containing all the possible elements of the matter to be formed. Of course, also $\mathcal{F}$ must be provided as an input to the algorithm. This constitutes what we call the **Molecular Pattern Formation** problem on grids (MPF for short).

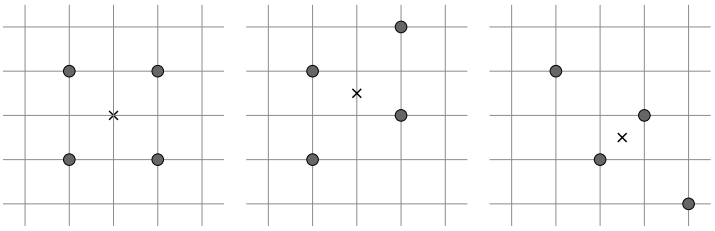

Fig. 4. Examples for the notion of center of a rotational configuration. From left to right, three rotational configurations C', C'', and C''' with $tc(C') = 1$, $tc(C'') = 2$, and $tc(C''') = 3$.

2.1. Formalization of the MPF problem

A square tessellation of the Euclidean plane is the covering of the plane using squares of side length 1, called tiles, with no overlaps and in which the corners of squares are identically arranged. Let S be the infinite lattice formed by the vertices of the square tessellation. The graph $G_{\square}$ is called **grid graph**, its vertices are the points in S and its edges connect vertices that are distance 1 apart.

Let $R = \{r_1, r_2, \dots, r_n\}$ be the set of robots. The topology where robots are placed is the grid graph $G_{\square} = (V, E)$. A function $\lambda : R \to V$ maps each robot to the vertex in $G_{\square}$ where the robot is placed. Assume that each robot knows the set of formable molecules $\mathcal{M}$ and the set $\mathcal{F}$ of possible patterns describing the matter to form. As said above, during the Look phase, each robot detects in the local coordinate system both the robots' positions and the set Mol of already formed molecules. We call $C = (G_{\square}, R, \lambda, Mol)$ a **configuration**. Notice that constraint C_1 imposes $Mol = \emptyset$ in each initial configuration.

Let C be a configuration and $\mathcal{A}$ be an algorithm processing C. By $C(t)$ we denote the configuration C observed by some robots at time $t \geq 0$ during the execution of $\mathcal{A}$. Formally, an **execution** of $\mathcal{A}$ from the initial configuration C is a sequence of configurations $\mathbb{E} : C(t_0), C(t_1), C(t_2), \dots$, where $\{t_i : i = 0, 1, \dots\}$ is the set of time instants at which at least one robot takes the snapshot $C(t_i)$ during its Look phase. With respect to the defined MOBLOT model, the **MPF** problem on the grid graph can be formalized as follows.

Definition 1. Given an initial configuration $C = (G_{\square}, R, \lambda, Mol = \emptyset)$, a set of formable molecules $\mathcal{M}$, and a set $\mathcal{F}$ of possible patterns describing the matter to form, the goal is to design a distributed algorithm $\mathcal{A}$ that works for each entity so that eventually they form some pattern in $\mathcal{F}$, if possible. Formally, $\mathcal{A}$ solves an instance $(C, \mathcal{M}, \mathcal{F})$ of the MPF problem for C if, for each possible execution $\mathbb{E} : C = C(t_0), C(t_1), \dots$ of $\mathcal{A}$, there exists a finite time instant $t_n > 0$ such that in $C(t_n)$ all robots have been assembled into molecules, the molecules form an element in $\mathcal{F}$, and no entity moves after t_n, i.e., $C(t_k) = C(t_n)$ for each $t_k \geq t_n$.

3. Necessary conditions on the feasibility of MPF

In this section, we state a necessary condition for the feasibility of MPF. As done above, we remind that the term **entity** is used when there is no need to distinguish among robots and molecules.

Two undirected graphs $G = (V, E)$ and $G' = (V', E')$ are *isomorphic* if there is a bijection φ from V to V' such that $\{u, v\} \in E$ if and only if $\{\varphi(u), \varphi(v)\} \in E'$. Vertices $u \in V$ and $v \in V'$ are equivalent if there exists an isomorphism φ such that $\varphi(u) = v$. An automorphism on a graph G is an isomorphism from G to itself, that is a permutation of the vertices of G that maps edges to edges and non-edges to non-edges. We denote by $\mathrm{Aut}(G)$ the set of all such automorphisms.

The concept of graph isomorphism can be extended to configurations in a natural way. Two configurations $C = (G_{\square}, R, \lambda, Mol)$ and $C' = (G'_{\square}, R', \lambda', Mol')$ are isomorphic if there exists an isomorphism φ between $G_{\square}$ and $G'_{\square}$ that can be extended to obtain a bijection from $V \cup R$ to $V' \cup R'$ that maps vertices into vertices and robots into robots and such that:

- two robots can be associated by φ only if they have the same color and reside on equivalent vertices, that is if $\varphi(r) = r'$ then $\varphi(\lambda(r)) = \lambda'(r')$;
- it preserves molecules: if $\mu = \{r_{i_1}, \dots, r_{i_t}\} \in Mol$ then $\{\varphi(r_{i_1}), \dots, \varphi(r_{i_t})\} = \mu' \in Mol'$ and $\mu = \mu'$, i.e., μ and μ' are of the same kind (i.e., same shape, same colored robots,...).

In this way, analogously to graph automorphism, an automorphism of a configuration C is an isomorphism from C to itself, and the set of all automorphisms of C forms a group under the composition operation that we call automorphism group of C and denote as $\mathrm{Aut}(C)$. Moreover, if $|\mathrm{Aut}(C)| = 1$ we say that C is **asymmetric**, otherwise it is **symmetric**. We have already defined the notion of equivalence between vertices. Concerning a configuration C, two distinct entities of C (robots or molecules) are **equivalent** if there exists $\varphi \in \mathrm{Aut}(C)$ that maps one into the other.

Remark 1. Let $C = (G_{\square}, R, \lambda, Mol)$ be a symmetric configuration, $\mathcal{A}$ be any algorithm acting on C, and E be any maximal subset of pairwise equivalent entities in C. Any move planned by $\mathcal{A}$ for an element of E applies to *all* set E.

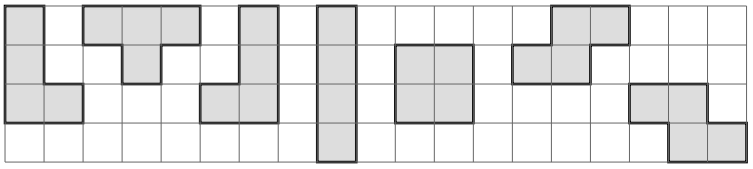

Fig. 5. All possible tetrominoes. According to their shape, they can be referred to as L, T, J, I, O, S, and Z, resp.

As chirality is assumed, it is easy to see that any configuration C defined on $G_{\square}$ admits one type of automorphisms only: **rotations**. A rotation is an isometry defined by a center c and a minimum angle of rotation $\alpha \in \{90, 180, 360\}$ working as follows: if the configuration is rotated around c by an angle α, then a configuration coincident with itself is obtained. The **order** of a configuration is given by $360/\alpha$. A configuration is **rotational** if its order is 2 or 4. The **type of center** of a rotational configuration C is denoted by $tc(C)$ and is equal to (cf. Fig. 4):

- 1, when the center of rotation is on a vertex of $G_{\square}$;
- 2, when the center of rotation is on a median point of an edge of $G_{\square}$;
- 3, when the center of rotation is on the center of a square of the tessellation forming $G_{\square}$.

The **symmetricity** of a configuration C, denoted as $\rho(C)$, is equal to its order unless its center is occupied by a robot, in which case $\rho(C) = 1$. It comes out that for the considered configurations defined on $G_{\square}$ we have $\rho(C) \in \{1, 2, 4\}$.

We defined $\rho()$ and $tc()$ for any configuration $C = (G_{\square}, R, \lambda, Mol)$ regardless whether Mol is empty or not. Concerning notation, we use $\rho(F)$ and $tc(F)$ to refer to any configuration forming a pattern $F \in \mathcal{F}$ (e.g., for the rotational pattern F in Fig. 3 we have $\rho(F) = 2$ and $tc(F) = 2$). As a special case, we use $\rho(\mu)$ and $tc(\mu)$ to refer to robots within a single molecule μ only. Moreover, for a given pattern $F \in \mathcal{F}$, let $Mol(F)$ denote the set of molecules that form F; clearly, each molecule in $Mol(F)$ also appears in $\mathcal{M}$. Our general result can be then formally stated:

Theorem 2. *Let $C = (G_{\square}, R, \lambda, Mol)$ be any configuration composed of* FSYNC *robots and $(C, \mathcal{M}, \mathcal{F})$ be an instance of the MPF problem. If there exists an algorithm $\mathcal{A}$ able to form a pattern $F \in \mathcal{F}$ from C, then one of the following holds:*

1. *$\rho(C)$ divides $\rho(F)$ and $(\rho(C) > 1 \Rightarrow tc(C) = tc(F))$;*
2. *$\exists\, \mu \in Mol(F)$: $\rho(C)$ divides $\rho(\mu)$ and $(\rho(C) > 1 \Rightarrow tc(C) = tc(\mu))$.*

Proof. We first assume that $\mathcal{A}$ can form F without moving any formed molecule. This implies that there exists a time $t_n > 0$ such that the disposal of the robots in $C(t_n)$ equals F and no molecule is moved in $C(t_i)$ for each $t_i < t_n$. In this case, we have from [16] that property (1) holds.

In what follows, we assume that $\mathcal{A}$ must move some molecules to form F. We also assume $\rho(C) > 1$, otherwise both properties (1) and (2) are trivially verified. Let $\mathbb{E} : C = C(t_0), C(t_1), \ldots$ be the execution of the algorithm $\mathcal{A}$. According to Remark 1, one or more sets of $\rho(C(t_0))$ pairwise equivalent robots move synchronously. Let $C(t_k)$, $k > 0$, be the first configuration containing molecules. If $C(t_k)$ contains more than one molecule, according to the synchronous moves and to the symmetricity of C, then (i) in $C(t_k)$ there is one or more sets of $\rho(C(t_0))$ molecules, (ii) in $C(t_k)$, the molecules in each set are all equivalent, (iii) $\rho(C(t_k))$ divides $\rho(F)$ and it is a multiple of $\rho(C(t_0))$, and (iv) the center of $C(t_0)$ and that of the configuration made by the formed molecules coincide. Then, from $C(t_k)$ on, each move planned by $\mathcal{A}$ maintains at least the same symmetricity $\rho(C(t_0))$ and the same type of center until F is formed. Then, $\rho(C(t_0))$ divides $\rho(F)$ and the center of the formed molecules is maintained. Summarizing, property (1) must hold.

If $C(t_k)$ contains just one molecule μ, then it must be formed around the center of the configuration so that $tc(\mu) = tc(C(t_0))$. Moreover, even in this case $\mathcal{A}$ makes $\rho(C(t_0))$ equivalent robots move synchronously, and hence $\rho(\mu)$ must be a multiple of $\rho(C(t_0))$. Summarizing, property (2) must hold. □

It is well known that the synchronization schedulers induce the following hierarchy (see, e.g. [30]): FSYNC robots are more powerful (i.e., they can solve more tasks) than SSYNC robots, that in turn are more powerful than ASYNC robots. As a consequence, any impossibility result stated for FSYNC robots also holds for SSYNC and ASYNC robots. This implies that the above result also holds when SSYNC and ASYNC robots are considered.

4. The Tetris-like MPF problem

In this section, we introduce **Tetris-Like MPF** (TL-MPF for short), a particular version of the MPF problem. TL-MPF will be used as a case study of the MOBLOT model in order to appreciate its facets in grids.

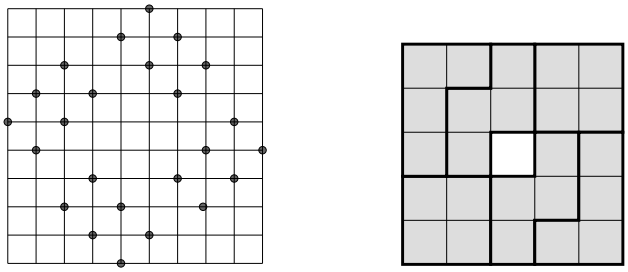

Fig. 6. (*left*) A configuration containing only robots and such that $\rho(C) = 4$. (*right*) A connected pattern F with $\rho(F) = 2$ (tetrominoes have a gray background color).

In TL-MPF all robots are identical (they are not colored), a molecule is formed by exactly four robots, and four robots form a molecule if they are disposed as a possible **tetromino** [26].[1] A tetromino is a geometric shape composed of four squares connected orthogonally (i.e., at the edges and not the corners). It follows that here $\mathcal{M}$ is composed by the seven possible tetrominoes, and each kind of tetromino is denoted by a single character among L, T, J, I, O, S, and Z, according to their shape, see Fig. 5. Concerning the matter to be formed, $\mathcal{F}$ is composed of just one pattern F – i.e., $\mathcal{F} = \{F\}$, where F is composed of four or more tetrominoes. Notice that, except the minimum number of molecules, there are no other restrictions on F and hence it may have any possible shape and may even have holes. According to the definition of F, the set of **initial** configurations consists of any configuration $C = (G_{\square}, R, \lambda, Mol = \emptyset)$ where R has a multiple of 4 and with at least 16 robots. We remark that we do not impose any further constraint on F, and hence F may have holes or may be disconnected.

According to the constraints on the molecule formation provided in Section 2, we know that robots forming any initial configuration C are pairwise non-adjacent, and any two tetrominoes formed from C by any algorithm cannot overlap. In the following, we say that two tetrominoes are **adjacent** when robots belonging to distinct tetrominoes are adjacent in $G_{\square}$.

We now specialize Theorem 2 to provide the definition of **potentially-formable patterns** from any initial configuration C of TL-MPF, i.e., patterns that respect the constraints dictated by Theorem 2 within TL-MPF. Note that, as C has an even number of robots, if $\rho(C) = 1$ then C is necessarily asymmetric. In fact, a robot occupying $c(R)$ leaves an odd number of robots to constitute a configuration with an even symmetricity, which is not possible. This remark implies that, in each initial configuration C for TL-MPF, the symmetricity induces a partition of all the entities in subsets having the following relevant properties: (1) each set has size equal to $\rho(C)$, and (2) in each set, the entities are pairwise equivalent. Each set in this partition is called **orbit**.

In principle, given an initial configuration C and a pattern F to be formed, it is possible that no algorithm exists for solving TL-MPF. According to $\rho(C)$, we have the following cases:

Corollary 3. *Given a configuration C and a pattern $F \in \mathcal{F}$, F is* potentially-formable *from C if one of the following conditions hold:*

1. $\rho(C) = 1$*;*
2. $\rho(C) = 2$ *and*
   (a) $tc(C) = tc(F)$ *and* $\rho(F) \in \{2, 4\}$*, or*
   (b) $tc(C) = 2$ *and* $\{\mathtt{S}, \mathtt{Z}, \mathtt{I}\} \cap Mol(F) \neq \emptyset$*, or*
   (c) $tc(C) = 3$ *and* $\mathtt{O} \in Mol(F)$*;*
3. $\rho(C) = 4$ *and*
   (a) $tc(C) = tc(F)$ *and* $\rho(F) = 4$*, or*
   (b) $tc(C) = 3$ *and* $\mathtt{O} \in Mol(F)$.

Proof. The statement follows directly from Theorem 2. In fact, as already observed, since C is a configuration defined on $G_{\square}$, we have $\rho(C) \in \{1, 2, 4\}$.

When $\rho(C) = 1$, then property (1) of Theorem 2 holds.

When $\rho(C) = 2$, then, according to the kind of molecules composing F, we have the three cases of the claim. The first dictated by property (1) of Theorem 2. The second and the third dictated by property (2) of Theorem 2, where $\{\mathtt{S}, \mathtt{Z}, \mathtt{I}\}$ are the only molecules with simmetricity equal to 2 whereas O is the only molecules with simmetricity equal to 4.

When $\rho(C) = 4$, then, according to the kind of molecules composing F, we have the two cases of the claim. The first dictated by property (1) of Theorem 2; the second dictated by property (2) of Theorem 2, where O is the only molecules with simmetricity equal to 4. □

[1] Notice that, in the remainder we use tetromino and molecule as synonyms.

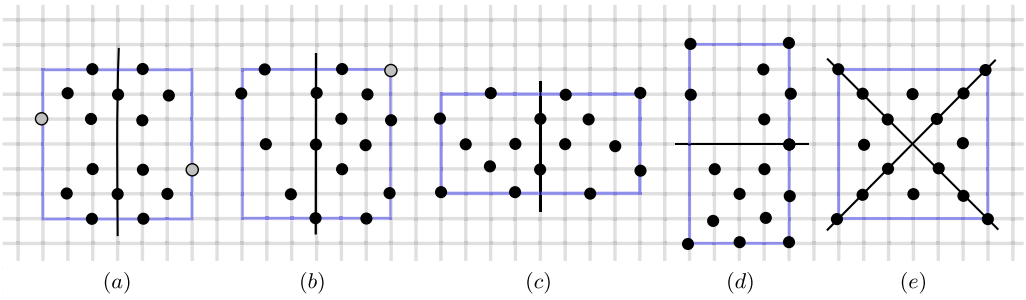

Fig. 7. Division of $mbr(R)$ into regions based on shape and $\rho(C)$. The $mbr(R)$ is shown in blue. Robots with the minimum view are shown in grey only when needed.

5. The algorithm $\mathcal{A}_{\mathsf{TL}}$

In this section, we describe an algorithm called $\mathcal{A}_{\mathsf{TL}}$ that solves TL-MPF for each pair (C, F), where C is an initial configuration composed of FSYNC robots and F is potentially-formable from C. This provides a complete characterization of the feasibility of TL-MPF. While reading this section, it might be helpful to refer to the running example provided in Section 6.

5.1. Concepts, definitions, and data structures used by $\mathcal{A}_{\mathsf{TL}}$

Let $C = (G_{\square}, R, \lambda, Mol)$ be an initial configuration. By $mbr(R)$, we denote the **minimum bounding rectangle** of R, that is the smallest rectangle (with sides parallel to the edges of $G_{\square}$) enclosing all robots (cf. Fig. 7). Note that $mbr(R)$ is unique. By $c(R)$, we denote the center of $mbr(R)$. Similarly, $mbr(F)$ is defined for the minimum bounding rectangle enclosing the molecules forming F.

In $\mathcal{A}_{\mathsf{TL}}$, robots encode the perceived configuration into a binary string called **lexicographically smallest string** and denoted as $LSS(R)$. It is computed by robots as follows (cf. [24]). Starting from the least significant bit, they assign a string to each corner of $mbr(R)$: the grid enclosed by $mbr(R)$ is analyzed row by row or column by column - the direction is given by the smallest side of $mbr(R)$ - and 1 or 0 correspond to the presence or the absence, resp., of a robot for each encountered vertex. From the 4 corners they get up to 8 different strings, and the lexicographically smallest one is $LSS(R)$. Note that if two strings obtained from opposite corners along opposite directions are equal, then the configuration is rotational, otherwise it is asymmetric.

The robot(s) with **minimum view** is the one with minimum position in $LSS(R)$ (as for the construction, the position is meant by scanning the string from the least significant bit). Fig. 7 can be used for providing examples about the view. In particular: in Fig. 7.(a) we have $\rho(C) = 2$, $LSS(R)$ = 0000100 0100010 1010101 0100010 1010101 0100010 0010000; in (b) we have $\rho(C) = 1$, $LSS(R)$ = 0000010 0001001 0100000 1001010 0010101 1001010 0100101. A similar approach can be used to associate a string made of letters to F (i.e., if the analyzed vertex is occupied by a robot forming molecule Z, then Z is inserted in the string, otherwise, if the vertex is unoccupied, X is inserted).

$\mathcal{A}_{\mathsf{TL}}$ assumes that robots are assigned to **regions** of $mbr(R)$ as follows (cf. Fig. 7). If $\rho(C) = 4$ then $mbr(R)$ is a square and hence it is partitioned by using two diagonals. If $\rho(C) = 2$ and $mbr(R)$ is a square, it is partitioned into 2 equal regions by a line passing through $c(R)$ and parallel to the sides of $mbr(R)$ where the robots with minimal view reside.[2] If $\rho(C) = 1$ and $mbr(R)$ is a square, it is partitioned into 2 equal regions by a line passing through $c(R)$ and parallel to the side of $mbr(R)$ where the robot with minimum view resides. If $\rho(C) = 1, 2$ and $mbr(R)$ is a rectangle, it is partitioned into 2 equal regions by a line passing through $c(R)$ and parallel to the shortest sides. Each robot is assigned (or belongs) to one of the formed regions, unless it is on a half-line of the lines used for partitioning $mbr(R)$; in this case, the robot belongs to the region to the right of the half-line. Each side ℓ of $mbr(R)$ entirely contained in a region is said to be "associated with" that region.

In Figs. 7.(a) and (b), the $mbr(R)$ is a square and it is partitioned into 2 equal regions by a line passing through $c(R)$ and parallel to the sides of $mbr(R)$ where the robots with minimum view reside; in Fig. 7.(c) we have $\rho(C) = 2$ while in Fig. 7.(d) we have $\rho(C) = 1$. However, in both the cases the $mbr(R)$ is a rectangle and it is partitioned into 2 equal regions by a line passing through $c(R)$ and parallel to the shortest sides; in Fig. 7.(e), we have $\rho(C) = 4$, $mbr(R)$ is a square, and hence it is partitioned by using two diagonals.

Two robots in C are called **partial-molecule** if they are adjacent (the term refers to the possibility that the robots can subsequently be joined by others until a molecule is obtained). Note that, by Property C_1, a partial-molecule cannot exist in initial configurations. Two robots in C are called **point-joined-robots** if they are aligned along the diagonal of a cell of the grid and their corresponding monominoes in the dual graph intersect in one point. We recall that Mol is part of the configuration and it is the set of all the molecules perceived by the robots during the Look phase; During the Look phase, robots perceive also $F' \subseteq Mol$, that is the set of formed molecules that are already assembled to form the matter. If the algorithm works correctly, it should be that F' is a sub-pattern of F.

[2] If a robot with minimum view is on a corner, it is assumed to reside on the clockwise side of $mbr(R)$.

Table 1
The decomposition of TL-MPF into tasks.

problem	*sub-problems*	*task*
TL-MPF	Making working Space	T_1
	Forming $\rho(C)$ new molecules	T_2
	Forming one central molecule	T_3
	Adding molecules to pattern	T_4
	Termination	T_5

5.2. On the structure of the algorithm

The algorithm $\mathcal{A}_{\mathsf{TL}}$ has been designed according to the methodology proposed in [31]. It is based on a preliminary decomposition approach: the problem is divided into a set of sub-problems so that each sub-problem is simple enough to be thought as a **task** that can be performed by (a subset of) entities. Table 1 (that will be better explained later) shows the decomposition into tasks for TL-MPF. More details will be given in Table 2. $\mathcal{A}_{\mathsf{TL}}$ is responsible for allowing entities to detect which task must be accomplished in any configuration observed during an execution. This is realized by assigning a **predicate** P_i to task T_i, for each i. The predicate is evaluated in the `Compute` phase on the basis of the view acquired during the `Look` phase. As soon as entities recognize that a task T_i must be accomplished, a move m_i – associated to that task, is performed by a subset of designed entities.

Predicates P_i are designed to guarantee the following required properties:

- Prop_1: *each P_i must be computable on the configuration perceived in each* `Look` *phase*;
- Prop_2: $P_i \wedge P_j = \mathtt{false}$, *for each $i \neq j$*;
- Prop_3: *for each possible perceived configuration there must exists a predicate P_i evaluated as true.*

If we guarantee that all these properties hold, then the algorithm can be used in the `Compute` phase as follows:

– if an entity executing the algorithm detects that predicate P_i holds, then it simply performs move m_i associated with T_i.

According to [31], in order to make Prop_2 valid, P_i is defined as follows:

$$P_i \equiv \mathtt{pre}_i \wedge \neg(\mathtt{pre}_{i+1} \vee \mathtt{pre}_{i+2} \vee \ldots \vee \mathtt{pre}_5) \tag{1}$$

where $\mathtt{pre}_i$ is the **precondition**[3] that characterizes T_i. Accordingly, in the `Compute` phase, each entity evaluates - with respect to the perceived configuration and the provided input - the preconditions starting from $\mathtt{pre}_5$ and proceeding in the reverse order until a true precondition is found. In case all predicates $P_5, P_4, \ldots, P_2$ are evaluated false, then task T_1, whose precondition is simply `true`, is performed. This guarantees that Prop_3 holds.

In the subsequent five sections, we give a description of each task T_i, by including also details about the corresponding move m_i and precondition $\mathtt{pre}_i$. The reader will be able to observe that, from the definition of the preconditions, it follows that also Prop_1 holds.

5.3. Tasks T_1 - making working space

The goal of this task is to increase the distance between robots. In fact, in an initial configuration, robots might be too close to each other (e.g., when robots occupy alternatively the vertices of the grid) and the movements might cause the formation of undesired partial-molecules.

During T_1, according to Remark 1, robots move away from $c(R)$. At the end of the task, consecutive orbits of robots (with respect to the distance from $c(R)$ of the robots composing the orbit) are at distance at least $\delta = 2$ from each other and there is also an empty space Q in the center of the configuration that contains at most the robots of the first orbit (the closest to $c(R)$). The space Q is a square centered in $c(R)$ and its side is $side(Q) = 2S$, where $S = \max\{w(F), h(F)\}$ and $w(F)$ and $h(F)$ are the width and the height, resp., of $mbr(F)$. The fixed distance δ guarantees that robots have enough space for moving and creating a molecule.

Within regions, robots are numbered as follows:

- Case $\rho(C) = 2, 4$. Let ℓ be the side of $mbr(R)$ associated with the region, and v be the leftmost vertex of ℓ.[4] Robots are numbered starting from v, proceeding along ℓ, then continuing in order with all the lines parallel to ℓ. By assuming that the region contains t robots, the first met robot is numbered as r^t and the remaining, in order, as $r^{t-1}, \ldots, r^1$. It is clear that the robots in a region all belong to different orbits and therefore the numbering of robots can be understood as a numbering for the orbits. Hence, orbits are denoted as $O^t, O^{t-1}, \ldots, O^1$.

[3] Specific conditions that must be verified in order to start a given task.
[4] Since robots share chirality, for each side they can distinguish between the two ending vertices: one on the left and one on the right.

- Case $\rho(C) = 1$. The two defined regions may have a different number of robots inside, say t_1 and t_2. Robots are numbered as in the previous cases in both the regions, but they are denoted as $\dot{r}^{t_1}, ..., \dot{r}^1$ in the region containing the robot with minimum view, and as $\ddot{r}^{t_2}, ..., \ddot{r}^1$ in the other region. Hence, orbits are denoted as $\dot{O}^{t_1}, \dot{O}^{t_1-1}, \ldots, \dot{O}^1$, and $\ddot{O}^{t_2}, \ddot{O}^{t_2-1}, \ldots, \ddot{O}^1$. Let $O^t, \ldots, O^1$, with $t = t_1 + t_2$, such that $O^t = \dot{O}^{t_1}, O^{t-1} = \ddot{O}^{t_2}$ and the remaining orbits $O^{t-2}, \ldots, O^1$ are defined by keeping orbits from the two regions in an alternating fashion as long as possible.

Let r be a robot in a region associated with a side ℓ, and assume $r \in O^i$: $cd_Q(O^i)$ represents the "current distance" of O^i from Q (that is the distance between r and the side of Q parallel to ℓ), it is negative if r is inside Q; $fd_Q(O^i)$ represents the "final distance" of O^i from Q, that is the distance that robots on O^i must have when the orbit is correctly positioned. These functions are formally defined as follows. When $\rho(C) = 2, 4$:

- $fd_Q(O^1) = \max\{S + 1, cd_Q(O^1)\}$
- $fd_Q(O^i) = \max\{fd_Q(O^{i-1}) + \delta, cd_Q(O^{i-1})\}, \forall i > 1$

When $\rho(C) = 1$:

- $fd_Q(O^1) = \max\{S + 1, cd_Q(O^1)\}$
- $fd_Q(O^i) = \max\{fd_Q(O^{i-1}) + \delta, cd_Q(O^{i-1})\}, \forall i > 1, i < t - 1$
- $fd_Q(O^t) = fd_Q(O^{t-1}) = \max\{fd_Q(O^{t-2}) + \delta, cd_Q(O^{t-2})\}$

The move planned for this task works as follows:

- Move m_1: *each robot in O^j, for each $j > 1$, moves perpendicularly to the side ℓ of $mbr(R)$ which it is associated to, increasing its distance from $c(R)$, until $cd_Q(O^j) = fd_Q(O^j)$.*

Note that the task makes all robots moving concurrently. By defining the two Boolean variables

- $P(k) \equiv$ *each orbit O^i, $i > k$, is correctly positioned with respect to $fd_Q()$;*
- $Q \equiv$ *square Q is formed with at most one orbit inside,*

it can be observed that task T_1 ends when both $P(1)$ and Q hold.

5.4. Task T_2 - forming $\rho(C)$ new molecules

The goal of this task is to create $\rho(C)$ new molecules to be added to the matter F' formed so far. Let B' be a Boolean variable that is true when one among the conditions 1, 2.a, and 3.a of Corollary 3 hold. Notice that in all such cases $\rho(F)$ is a multiple of $\rho(C)$, and when $\rho(C) > 1$ then C and F have the same type of center. Another condition that must be satisfied is that $|Mol \setminus F'| = 0$, i.e., there are no molecules already formed that must be moved.

In order to be executed, T_2 requires that B' holds and T_1 is completed (i.e., $P(1)$ and Q hold). If *ParMol* denotes the number of partial-molecules formed, then the precondition of T_2 is the following:

- $\texttt{pre}_2 \equiv B' \wedge |Mol \setminus F'| = 0 \wedge ((P(1) \wedge Q) \vee ParMol = \rho(C))$.

In this task, a relevant issue is that robots have to agree on which molecule μ in F must be formed (in $\rho(C)$ copies).

Definition 2 *(Disassembling sequence).* Let F be a pattern and ℓ be a side of $mbr(F)$ encoded with the minimal string within $LSS(F)$. Perform the following iterative process: (1) mark all molecules in F and create an empty ordered list $DS(F)$, (2) with respect to the marked molecules only, compute the set E of all the molecules that can be "extracted" from F through ℓ[5]; (3) insert in $DS(F)$ one of the molecules $\mu \in E$ including a robot of minimum view, (4) unmark all the molecules belonging to the same orbit of μ, (5) iterate from (2) as long as marked orbits exist. The order of the elements belonging $DS(F)$ constitutes a disassembling sequence for F.

For instance, $DS(F) = (\texttt{J}, \texttt{O}, \texttt{Z})$ for the pattern F shown in Fig. 6 (where ℓ coincides with the left and right sides). The algorithm selects the molecule μ to build by comparing the formed sub-pattern F' with F. See also Fig. 11. According to this comparison, the algorithm searches for molecules μ' and μ'' in F' having minimum and maximum positions in $DS(F)$, resp.; if μ'' is not the last element in $DS(F)$, then μ is the next to μ'' in $DS(F)$, otherwise it precedes μ' in $DS(F)$. In this way, the disassembling sequence in $DS(F)$ is used to correctly compose the pattern. See the running example in Section 6 and in particular Fig. 11 in which the first molecule formed is necessarily $\texttt{O}$.

[5] I.e., when the molecule's projection on ℓ is not obstructed by any other molecule.

Let O_1^*, O_2^*, O_3^*, O_4^* be the consecutive orbits closest to $c(R)$ and containing robots not involved in any molecule. The move planned for this task works as follows:

- Move m_2:
  - *If no partial-molecule is formed, then each robot in O_1^* moves toward the robot in O_2^* belonging to the same region to form a partial-molecule,*
  - *else, each robot closest to $c(R)$, excluding those forming the partial-molecule, i.e. the robots of the former orbits O_3^* or O_4^*, moves toward the partial-molecule within the same region toward a position adjacent to the partial-molecule and according to molecule μ to be formed.*

The configuration obtained after task T_2 contains $\rho(C)$ **new molecules**.

5.5. Tasks T_3 - forming one central molecule

This task processes the configurations of robots fulfilling one among conditions 2.b, 2.c, and 3.b of Corollary 3, according to the pattern F that must be potentially-formable from C.

This task is alternative to T_2 as it builds a single molecule μ in the center of C, usually when it is required to break the symmetry by means of a molecule. Let B be a Boolean variable that it is true if one among the conditions 2.b, 2.c, and 3.b from Corollary 3 holds. Note that T_3 activates only when the square Q is formed. Let PJR be a boolean variable that is true if there exist two point-joined-robots. In particular, the precondition of T_3 is equal to:

- $\texttt{pre}_3 \equiv B \wedge ((P(1) \wedge |Mol| = 0) \vee (P(2) \wedge (ParMol = 1 \vee PJR)))$.

Robots must agree on which molecule in F must be formed as first. It corresponds to the molecule fulfilling conditions 2.b, 2.c, and 3.b of the definition of potentially-formable pattern in Corollary 3 and with the highest position on the disassembling sequence of F. Four robots belonging to one or two orbits O_1, O_2 (according whether $\rho(C) = 4$ or $\rho(C) = 2$, resp.) closest to $c(R)$ are selected. μ is embedded on the grid so that the center of the molecule coincides with $c(R)$. See Fig. 9 and the running example in Section 6.

The move planned for this task works as follows:

- Move m_3:
  - *if $\rho(C) = 2$, then first robots in O_1 move toward $c(R)$ until forming a partial-molecule when $tc(C) = 2$ and $\mu \in \{\mathtt{S}, \mathtt{Z}, \mathtt{I}\}$, or reaching the minimum distance between them when $tc(C) = 3$ and $\mu = \mathtt{O}$. Then robots in O_2 move toward robots in O_1 by forming the desired molecule μ;*
  - *if $\rho(C) = 4$, robots in O_1 move toward $c(R)$ until forming $\mu = \mathtt{O}$.*

The task ends when all the 4 moving robots reach their targets and the molecule is built with its center in $c(R)$. At the end of task T_3, a new configuration C' is obtained with $\rho(C') = 1$.

5.6. Task T_4 - adding molecules to pattern

During this task, the molecules built during T_2 or T_3 move to start forming F or to be aggregated to F' (created by previous executions of this task). We define **quadrant** any of the four areas into which the square Q is divided by two orthogonal lines parallel to the sides of Q and intersecting in $c(R)$. To test if the pattern creation has already started, robots check whether (1) there exists a sub-pattern F' in one of the four quadrants, or (2) there exists a sub-pattern F' embedded so that it is centered in $c(R)$. This allows robots to evaluate the following precondition of T_4:

- $\texttt{pre}_4 \equiv \ |Mol \setminus F'| > 0$.

During this task, $\rho(C)$ can be 1, 2 or 4. To correctly determine the move to be performed, the algorithm considers four disjoint cases, which are defined according to $\rho(C)$, F', and Mol. Hence, the move planned for this task works as follows:

- Move m_4:
  - Case 1: $\rho(C) = 1$, $|F'| = 0$, $Mol = \{\mu\}$, *and μ is centered in $c(R)$. It is clear that the current configuration C has been created in T_3. In this case, m_4 breaks the symmetry by simply moving μ away from the center in an arbitrary direction. Notice that, in the formed configuration C' we have $\rho(C') = 1$, $|F'| = 0$, $Mol = \{\mu\}$, and μ is no longer centered in $c(C')$.*
  - Case 2: $\rho(C) = 1$, $|F'| = 0$, $Mol = \{\mu\}$, *and μ is not centered in $c(R)$. The current configuration has been created in T_4 (Case 1), or by T_2. F is meant to be embedded in the quadrant q of Q closest to μ, and m_4 moves μ toward the position in the embedded F corresponding to the minimum position of its shape in the disassembling sequence $DS(F)$. Concerning how F is embedded into q: let ℓ be a side of $mbr(F)$ used in the disassembling sequence (cf. Definition 2), and let c be the corner of ℓ with larger label; c is mapped on the vertex in q closest to $c(R)$, and ℓ is mapped on the counter-clockwise internal side of q. Notice that this embedding is used whenever the configuration is asymmetric.*

Table 2
Tables formalizing the most important details about $\mathcal{A}_{\mathsf{TL}}$.

var	*definition*
$P(k)$	each orbit O^i, $i > k$, is correctly positioned with respect to $fd_Q()$
Q	square Q is formed with at most one orbit inside
B	one among the conditions 2.b, 2.c, and 3.b of Corollary 3 holds
B'	one among the conditions 1, 2.a, and 3.a of Corollary 3 holds
PJR	there exist two point-joined-robots

(a) The basic Boolean variables used to define all the tasks' preconditions. They are formally introduced along Sections 5.3-5.7.

pre	*definition*
$\mathtt{pre}_1$	`true`
$\mathtt{pre}_2$	$B' \wedge \lvert Mol \setminus F' \rvert = 0 \wedge ((P(1) \wedge Q) \vee ParMol = \rho(C))$
$\mathtt{pre}_3$	$B \wedge ((P(1) \wedge \lvert Mol \rvert = 0) \vee (P(2) \wedge (ParMol = 1 \vee PJR)))$
$\mathtt{pre}_4$	$\lvert Mol \setminus F' \rvert > 0$
$\mathtt{pre}_5$	F is formed

(b) Tasks' preconditions formulated according to the Boolean variables in Table 2a. We recall that $ParMol$ denotes the number of partial-molecules formed (cf. Section 5.4) and $F' \subset Mol$ is the set of molecules already formed and observed by robots during the `Look` phase.

problem	*sub-problems*	*task*	*move*	*transitions*
TL-MPF	Making working Space	T_1	m_1	T_2, T_3
	Forming $\rho(C)$ new molecules	T_2	m_2	T_4
	Forming one central molecule	T_3	m_3	T_4
	Adding molecules to pattern	T_4	m_4	T_2, T_5
	Termination	T_5	m_5	T_5

(c) The decomposition of TL-MPF into tasks exploited by $\mathcal{A}_{\mathsf{TL}}$. The last column shows the transitions among tasks generated by the provided algorithm. Details about moves $m_1, \ldots, m_5$ are provided in Sections 5.3-5.7.

- Case 3: $\rho(C) = 1$ *and* $|F'| > 0$. *In this case, there exists only one molecule* μ *which is not part of* F'. *This is due to the fact that molecules are assembled one at time since* $\rho(C) = 1$. *Move* m_4 *makes* μ *moving toward its target identified by comparing* F' *with the position of* μ *in the disassembling sequence of* F.
- Case 4: $\rho(C) > 1$ *and* $|Mol \setminus F'| = \rho(C)$. *In this case,* F' *is embedded so that it is centered in* $c(R)$. *There are exactly* $\rho(C)$ *molecules which are not part of* F', *and they must be moved toward their final targets. The final targets are obtained by comparing* F *with* F'. *During the movements, each molecule remains in the same region. The last time this task is executed,* F *is finally formed.*

5.7. Task T_5 - termination

It refers to the **termination** problem, that is the task in which each entity recognizes that the pattern is formed. The precondition is the following:

- $\mathtt{pre}_5 \equiv$ *"F is formed"*.

When this task is recognized, each entity maintains the current position and hence m_5 corresponds to the nil movement.

5.8. Formalization of $\mathcal{A}_{\mathsf{TL}}$

The previous five sections have provided a detailed description for each task T_i in which the TL-MPF problem has been decomposed. They also include details about the corresponding move m_i and precondition $\mathtt{pre}_i$, $1 \leq i \leq 5$. Table 2 summarizes all the Boolean variables used to define the tasks' preconditions and all the preconditions, respectively, for such tasks. A predicate $P_i = \mathtt{pre}_i \wedge \neg(\mathtt{pre}_{i+1} \vee \mathtt{pre}_{i+2} \vee \ldots \vee \mathtt{pre}_5)$ is assigned to each task T_i. According to this definition of the predicates, $\mathcal{A}_{\mathsf{TL}}$ works as follows: in the `Compute` phase, each entity evaluates - with respect to the perceived configuration and the provided input - the preconditions starting from $\mathtt{pre}_5$ and proceeding in the reverse order until a true precondition is found.

6. Running example

In this section, we show the effectiveness of algorithm $\mathcal{A}_{\mathsf{TL}}$ with an extended example. In particular, we simulate the running of $\mathcal{A}_{\mathsf{TL}}$ by starting from the configuration C_1 shown in Fig. 8.left. Notice that in what follows, when a predicate P_i holds with respect to the current configuration C, we say that C *is in* T_i.

As $\rho(C) = 4$ in C_1, Fig. 8.right shows the subdivision into four regions and the subset of six robots belonging to one region.

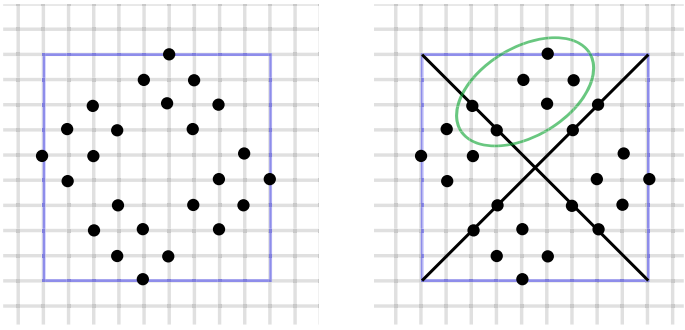

Fig. 8. Configuration C_1 in T_1 and its subdivision into regions.

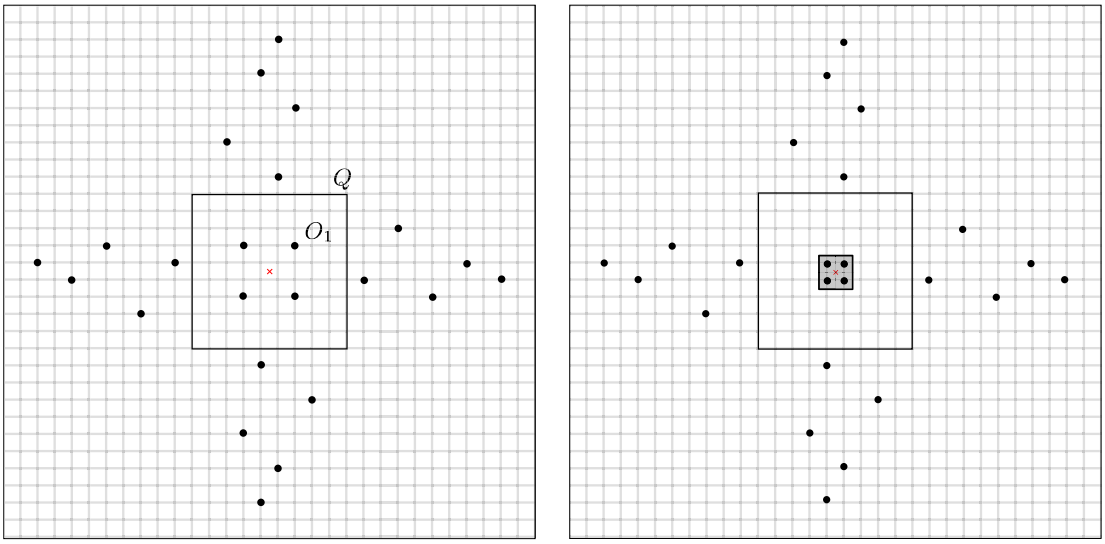

Fig. 9. *(left)* Configuration C_2; *(right)* Configuration C_3.

Since in C there are no molecules formed, then $\texttt{pre}_5$ and $\texttt{pre}_4$ are false. $P(1)$ is false and there is no partial-molecule or point-joined robots so $\texttt{pre}_3$ is false. Since B' is false, then $\texttt{pre}_2$ is false. As $\texttt{pre}_1 = \texttt{true}$, by Equation (1), then $P_1 = \neg(\texttt{pre}_2 \vee \texttt{pre}_3 \vee \texttt{pre}_4 \vee \texttt{pre}_5)$ and hence C_1 is in T_1.

Move m_1 is designed to make both $P(1)$ and Q true. Since $\rho(C) = 4$, m_1 moves all the robots but the 4 closest to $c(R)$ so as to enlarge the $mbr(R)$. This step ensures that robots have enough space to form the designed molecules in the successive tasks. Actually, from C_1, both $P(1)$ and Q becomes true as soon as configuration C_2 is achieved, where exactly 4 robots remain inside the square Q (cf. Fig. 9). When this happens, still no molecules are yet formed, that is $\texttt{pre}_5$, $\texttt{pre}_4$ are false. However $\texttt{pre}_3$ is true: B is true since condition 3.b of Corollary 3 holds and $|Mol| = 0$, hence C_2 belongs to T_3.

Move m_3 moves 4 robots inside Q to form the first molecule O in the center of the configuration. Once this happens, C_3 is obtained, $\texttt{pre}_5$ is still false, $|Mol \setminus F'| = 1$, $\texttt{pre}_4$ becomes true and hence the configuration is in T_4.

Notice that move m_4 is defined by cases. Since in C_3 we have that $\rho(C) = 1$, $|F'| = 0$, $Mol = \{\mu\}$, and μ is centered in $c(R)$, then Case 1 applies. As a consequence, m_4 breaks the symmetry by moving the molecule away from the center in an arbitrary direction. The obtained configuration is still in T_4, but now Case 2 applies. In fact, $\rho(C_4) = 1$, $|F'| = 0$, $Mol = \{\mu\}$, and μ is not centered in $c(R)$. In this case, F is meant to be embedded in the quadrant of Q closest to μ, and m_4 moves μ toward the position in the embedded F corresponding to the minimum position of its shape in the disassembling sequence $DS(F)$. It is easy to observe that it is sufficient a single LCM cycle to move the molecule in its final destination, as shown in Fig. 10.left. The obtained configuration is denoted as C_4 and it is still in T_4, but now Case 3 applies.

In C_4, clearly $\texttt{pre}_5$ is false, $|Mol \setminus F'| = 0$ hence $\texttt{pre}_4$ is false. As $\rho(C) = 1$, B is false and so $\texttt{pre}_3$ is, while B' holds since condition 1 of Corollary 3 holds. Both Q and $P(1)$ hold, so the configuration is in T_2.

Four consecutive orbits of robots are selected, $O_1^*, O_2^*, O_3^*, O_4^*$. Since $\rho(C) = 1$ there is only one robot per orbit. Hence, the robot in O_1^* moves toward the robot in O_2^* belonging to the same region to form a partial-molecule and the configuration is still in T_2. Then, the second condition of move m_2 holds, so the robots closest to $c(R)$, excluding those forming the partial-molecule, move toward the partial-molecule within the same region in a position adjacent to the partial-molecule (according to molecule μ to be formed). According to Definition 2, the next molecule to be formed is a Z.

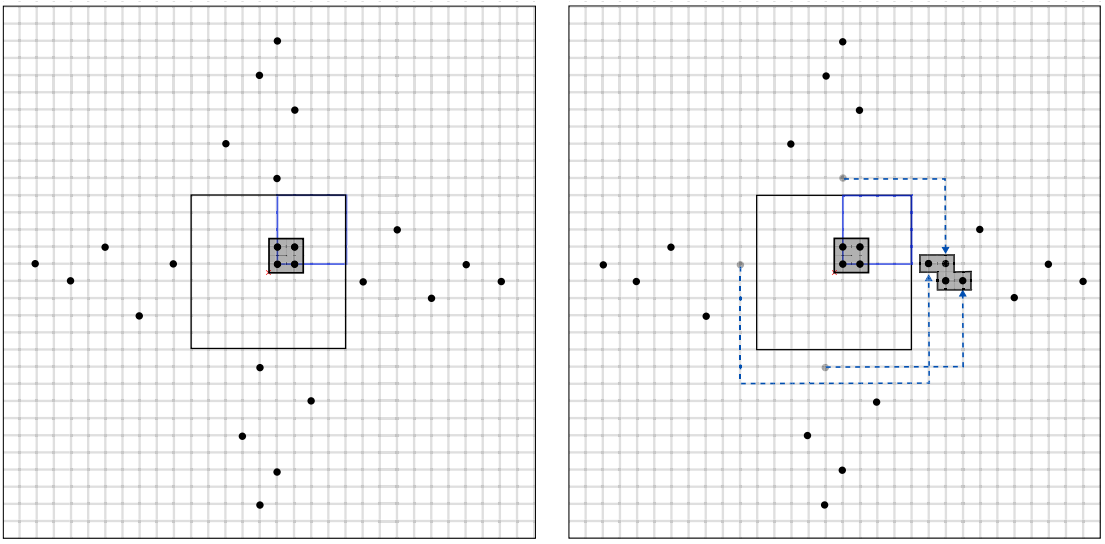

Fig. 10. *(left)* Configuration C_4 after T_4; *(right)* Configuration C_5.

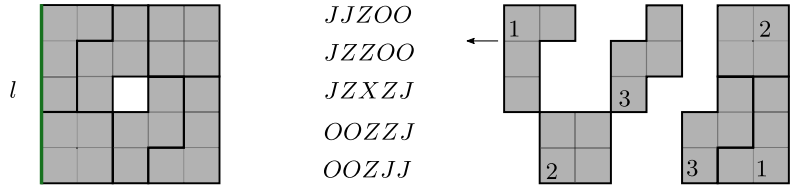

Fig. 11. From left: the pattern F and the side l; the view of robots; the disassembly sequence. The assembly sequence is OZJ for the first region, ZOJ for the second one.

Fig. 11.right shows the assembling and disassembling sequence of the two regions of the pattern F (Fig. 11.left), according to the minimal string associated with F (Fig. 11.middle). In particular the assembly sequence of the first region is OZJ because the algorithm was forced to create a molecule O as first, whereas the sequence for the second region is ZOJ.

The formation of molecule Z is done as shown in Fig. 10.right, where configuration C_5 is represented.

In C_5, $\texttt{pre}_5$ is false, but $\texttt{pre}_4$ is true. In fact, in T_4 - Case 3 - the algorithm moves the new molecule created by T_2 toward F', positioning it accordingly to the embedding of F. Once this happens, a new configuration belonging to T_2 is obtained, see C_6 in Fig. 12.left, hence the execution of the algorithm cycles among T_2 and T_4 until pattern F is formed, i.e. configuration C_7, shown in Fig. 12.right is achieved, where clearly $\texttt{pre}_5$ holds.

7. Correctness

In this section, we formally prove that algorithm $\mathcal{A}_{\mathsf{TL}}$ solves the TL-MPF problem. According to the methodology proposed in [31], the correctness of the proposed algorithm can be obtained by proving that all the following properties hold:

- H_1: *The algorithm never generates unsolvable configurations. According to Corollary 3, this implies that each configuration $C(t)$, $t > 0$, generated by the algorithm is potentially-solvable.*
- H_2: *The movement of each robot is collision-free.*
- H_3: *For each task T_i, the transitions from T_i to any other task are exactly those declared in Table 1.*
- H_4: *Each transition in Table 1 occurs after a finite number of cycles. This means that the generated configurations can remain in the same task only for a finite number of cycles.*

Since these properties must be proved for each transition/move, then in the following we provide a specific lemma for each task. Notice that Property H_3 does not directly imply that robots/molecules "complete" each task in a finite amount of time. In fact, there is a cycle created by transitions between tasks T_2 and T_4. Anyway, a final theorem will assess the correctness of $\mathcal{A}_{\mathsf{TL}}$ by making use of all the proved properties H_1–H_4 for each task and by also showing that there is a finite number of transitions between tasks T_2 and T_4.

Lemma 4. *Let C be a configuration in T_1. From C, $\mathcal{A}_{\mathsf{TL}}$ eventually leads to a configuration belonging to T_2 or T_3.*

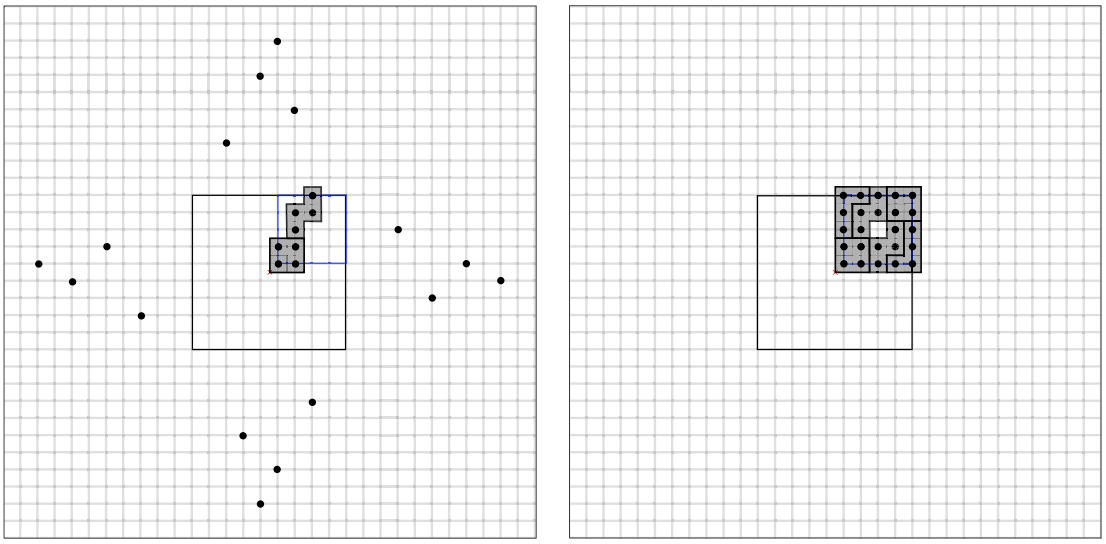

Fig. 12. *(left)* Configuration C_6; *(right)* Configuration C_7.

Proof. During task T_1 robots move away from $c(R)$ to leave a square Q of side $2S$ centered in $c(R)$ with at most one orbit inside, while leaving at least δ space between consecutive orbits. Let us analyze properties H_i, for $1 \leq i \leq 4$, separately.

H_1: By means of move m_1, all the orbits but O^1 move farthest from $c(R)$. During the movements, robots move synchronously keeping the same symmetricity of the initial configuration and the same type of center. The final targets $fd_Q()$ reached by the robots are defined so that each orbit is at a different distance from the center $c(R)$. Due to the synchronicity of the movements, $\rho(C)$ is maintained the same along all the movements and after the robots reach their targets.

H_2: During task T_1, robots increase their distance from $c(R)$ and from other robots moving in a perpendicular direction with respect to the side of $mbr(R)$ to which they are associated with. Orbits move all together and the targets for robots are defined so that the distance between consecutive orbits is at least δ therefore collisions cannot occur during the movements of the robots.

H_3: We show that each configuration generated from C remains in T_1 until all robots reach their targets. When T_1 starts, no molecules are yet formed therefore $\texttt{pre}_5$ is false. During the movements of the robots, no molecules are built so $\texttt{pre}_4$ remains false and the configuration does not belong to T_4. Therefore at the end of T_1 the configuration is either in T_2 or in T_3.

H_4: As long as the configuration remains in T_1, the distance of each moving robot from its target decreases. Hence, the task ends within a finite number of computational cycles. □

Lemma 5. *Let C be configuration in T_2. From C, $\mathcal{A}_{\mathsf{TL}}$ eventually leads to a configuration belonging to T_4.*

Proof. During task T_2, $\rho(C)$ new molecules are built. Let us analyze properties H_i, for $1 \leq i \leq 4$, separately.

H_1: During this task, the algorithm moves $\rho(C)$ orbits of robots and builds $\rho(C)$ molecules synchronously working by regions. Hence the symmetricity of the configuration is kept during the execution of T_2. If there are only four robots left and they move to form the last molecule, at least two of these four robots are on the sides of $mbr(R)$ holding its shape. When the two outermost robots are on $mbr(R)$, both $mbr(R)$ and the type of center do not change during the movements of O_1^* toward O_2^*. When all the four robots are on $mbr(R)$, then the shape of $mbr(R)$ and the type of center change during the movement of O_1^*. However the type of center changes only either horizontally or vertically, never in both directions so it can never coincide with $tc(F)$, eventually changing $\rho(C)$. So $c(R) \neq c(F)$ during all the execution of T_2 and $\rho(C)$ is kept until the end of the task.

H_2: During task T_2 the four orbits of robots O_1^*, O_2^*, O_3^*, O_4^* that are the closest to $c(R)$ are selected to move. One robot for each region moves at a time. Firstly the robots from O_1^* move towards the robots in O_2^*. Note that, at the end of task T_1 the distance between consecutive orbits is at least δ, so the robots in O_1^* move in an empty space until they form a partial-molecule with the robots of orbit O_2^*. The partial-molecule is built outside Q. Since O_3^*, O_4^* are the orbits of robots closest to $c(R)$ not involved in any molecule, they move toward the partial-molecules without any collision with other robots. Their target is a position adjacent to the partial-molecule according to the molecule μ.

H_3: We show that each configuration generated from C remains in T_2 until $\rho(C)$ molecules are built. Since the configuration is in T_2, all the preconditions $\texttt{pre}_i$ with $i > 2$ are false. In fact F is not formed so $\texttt{pre}_5$ is false, and since there are no

molecules that are not part of the pattern until T_2 ends, then $\texttt{pre}_4$ is false. During task T_2, $\rho(C)$ can be 1, 2, or 4. If $\rho(C) = 1$ then one molecule is built and the movements of the robots never augment the symmetricity of the configuration (see Property H_1 above) hence variable B belonging to $\texttt{pre}_3$ is always false. If in T_2, $\rho(C) = 2, 4$ then condition 2 of Corollary 3 is false and the movements of the robots during move m_2 cannot change the symmetricity of the configuration nor the type of center. Therefore variable B' is false and does not change its value during the execution of T_2, so $\texttt{pre}_3$ is false.

H_4: As long as the configuration remains in T_2, the distance of each moving robot from their target decreases. Hence, within a finite number of computational cycles the task ends and the configuration is in T_4. □

Lemma 6. *Let R be a configuration in T_3. From C, $\mathcal{A}_{\mathsf{TL}}$ eventually leads to a configuration belonging to T_4.*

Proof. Let us analyze properties H_i, for $1 \leq i \leq 4$, separately.

H_1: During this task $\rho(C) = 2, 4$ and one among conditions 2.b, 2.c, 3.b of Corollary 3 holds. One or two orbits of robots closest to $c(R)$ move. During the movements of $\rho(C)$ robots towards $c(R)$, the symmetricity of C and the type of center cannot change due the synchronicity of the moves, until the task is over. When the task ends, a new configuration C' is obtained such that $\rho(C') = 1$. Condition 1 of Corollary 3 holds and the configuration is still potentially-formable.

H_2: Task T_3 starts after T_1, that is square Q centered in $c(R)$ with at most four robots inside has been realized. The four robots closest to $c(R)$ move toward $c(R)$ to build the first molecule. They belong to one or two orbits depending on $\rho(C)$. There are no other robots between these four and $c(R)$, so no collision can occur. As soon as they become adjacent, they stop and the molecule is formed.

H_3: The precondition $\texttt{pre}_4$ remains false until the molecule is built. As soon as the molecule is completed, precondition $\texttt{pre}_4$ becomes true and the configuration is in T_4.

H_4: As long as the configuration remains in T_3, the distance of each moving robot from $c(R)$ decreases. Hence, within a finite number of computational cycles, the robots create a molecule and the configuration is in T_4. □

Lemma 7. *Let C be a configuration in T_4. From R, $\mathcal{A}_{\mathsf{TL}}$ eventually leads to a configuration belonging to T_2 or T_5.*

Proof. During this task, the molecules formed during either task T_2 or T_3 move to join the pattern F'. Let us analyze properties H_i, for $1 \leq i \leq 4$, separately.

H_1: If $\rho(C) = 1$, the pattern F is embedded in a quadrant q of Q. One molecule goes toward its target in q and $\rho(C)$ cannot increase during the movement. Then, the configuration remains potentially-solvable by Corollary 3. If $\rho(C) = 2, 4$, during the movements of the molecules, $c(R)$ does not change and so does $tc(C)$, so the conditions 2 and 3 of Corollary 3 still hold.

H_2: During this task only molecules move, therefore collisions between robots cannot happen. When $\rho(C) = 2, 4$, there is one molecule in each region that goes toward F' that is embedded in the center of $c(R)$. The space between the molecules μ and F' is empty and the molecules move on free trajectories. When $\rho(C) = 1$ then one molecule goes toward F' that is embedded in a quadrant q of Q. The quadrant q is big enough to contain F, the space between F' and the molecule μ is empty and the molecule moves on a free trajectory. Therefore no collision can occur. Note that, as more robots are assembled into molecules the empty space around Q enlarges. Moreover the disassembly sequence ensures that molecule can set in place in F without colliding with other molecules.

H_3: Precondition $\texttt{pre}_4$ remains true until there are molecules that are not yet part of the pattern F'. If there are no robots left then, as soon as the molecules join the pattern F', the pattern F is completed and the configuration is in T_5, otherwise the configuration is in T_2.

H_4: As long as the configuration remains in T_4, the distance of each moving molecule from F' decreases. Hence, within a finite number of computational cycles, the molecules join the pattern. □

Theorem 8. *$\mathcal{A}_{\mathsf{TL}}$ solves the TL-MPF problem for any initial configuration C and any pattern F if and only if F is potentially-formable from C.*

Proof. Lemmata 4-7 ensure that properties H_1, H_2, H_3, and H_4 hold for each task $T_1, T_2, \ldots, T_5$. This implies the following properties: F is always potentially-solvable; the movements of the robots and molecules are all collision-free; all the transitions are those reported in Table 2c; and the generated configurations can remain in the same task only for a finite number of cycles. Lemmata 4-7 also show that from a given task only subsequent tasks can be reached, or $\texttt{pre}_5$ eventually holds (and hence TL-MPF is solved). The only exception is the cycle among tasks T_2 and T_4. However, in this case, at the end of T_4, the number of molecules composing the pattern increases, and since no molecule is moved away from the pattern, task T_5 is reached from T_4 after a finite number of transitions between T_2 and T_4. This formally implies that, for each initial configuration C and for each execution $\mathbb{E} : C = C(t_0), C(t_1), C(t_2), \ldots$ of $\mathcal{A}_{\mathsf{TL}}$, there exists a finite time $t_j > 0$ such that the disposal of the entities in $C(t_j)$ equal the pattern to be formed in the TL-MPF problem and $C(t_k) = C(t_j)$ for each time $t_k \geq t_j$. □

8. Conclusion

In this paper, we have extended the recently introduced MOBLOT model to the case in which robots and molecules move along the edges of a grid graph. We have also formalized the Molecular Pattern Formation (MPF) problem where the final configuration is composed by molecules only. MPF is an extension of the well-studied Pattern Formation problem defined in the OBLOT model. For MPF, we have proven a necessary condition for its resolution. Finally, we have introduced TL-MPF, a specialized version of MPF called Tetris-like MPF in which the formable molecules correspond to the set of seven tetrominoes. For TL-MPF, we have given a complete characterization, providing a distributed algorithm able to form a molecular pattern whenever the necessary condition for the solvability of MPF is verified.

There are many directions for future research in the proposed model. The most natural one is to investigate about a (complete) characterization on the solvability of the general MPF problem according to the assumed capabilities for robots and molecules. Others concern the formalization of possible self-reconfigurable matter problems, as well as problems related to the matter movement. Completely different tasks/problems can also be thought, perhaps by adding further capabilities to the molecules.

CRediT authorship contribution statement

Serafino Cicerone: Conceptualization, Data curation, Formal analysis, Funding acquisition, Investigation, Methodology, Project administration, Resources, Software, Supervision, Validation, Visualization, Writing – original draft, Writing – review & editing. **Alessia Di Fonso:** Conceptualization, Data curation, Formal analysis, Funding acquisition, Investigation, Methodology, Project administration, Resources, Software, Supervision, Validation, Visualization, Writing – original draft, Writing – review & editing. **Gabriele Di Stefano:** Conceptualization, Data curation, Formal analysis, Funding acquisition, Investigation, Methodology, Project administration, Resources, Software, Supervision, Validation, Visualization, Writing – original draft, Writing – review & editing. **Alfredo Navarra:** Conceptualization, Data curation, Formal analysis, Funding acquisition, Investigation, Methodology, Project administration, Resources, Software, Supervision, Validation, Visualization, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

References

[1] S. Cicerone, A. Di Fonso, G. Di Stefano, A. Navarra, Molecular robots with chirality on grids, in: Algorithmics of Wireless Networks - 18th International Symposium on Algorithmics of Wireless Networks, ALGOSENSORS 2022, in: LNCS, vol. 13707, Springer, 2022, pp. 45–59.
[2] A.L. Christensen, Self-reconfigurable robots - an introduction, Artif. Life 18 (2) (2012) 237–240, https://doi.org/10.1162/artl_r_00061.
[3] M. Amir, A.M. Bruckstein, Minimizing travel in the uniform dispersal problem for robotic sensors, in: Proc. 18th Int.'l Conf. on Autonomous Agents and MultiAgent Systems (AAMAS), International Foundation for Autonomous Agents and Multiagent Systems, 2019, pp. 113–121.
[4] E. Sahin, Swarm robotics: from sources of inspiration to domains of application, in: E. Sahin, W.M. Spears (Eds.), Swarm Robotics, SAB 2004 Int.'l Workshop, Santa Monica, CA, USA, July 17, 2004, Revised Selected Papers, in: LNCS, vol. 3342, Springer, 2004, pp. 10–20.
[5] M. Rubenstein, C. Ahler, R. Nagpal, Kilobot: a low cost scalable robot system for collective behaviors, in: IEEE Int.'l Conf. on Robotics and Automation (ICRA), IEEE, 2012, pp. 3293–3298.
[6] Z. Derakhshandeh, R. Gmyr, T. Strothmann, R.A. Bazzi, A.W. Richa, C. Scheideler, Leader election and shape formation with self-organizing programmable matter, in: A. Phillips, P. Yin (Eds.), DNA Computing and Molecular Programming - 21st International Conference, DNA 21, Boston and Cambridge, MA, USA, August 17-21, 2015. Proceedings, in: LNCS, vol. 9211, Springer, 2015, pp. 117–132.
[7] J.J. Daymude, K. Hinnenthal, A.W. Richa, C. Scheideler, Computing by programmable particles, in: P. Flocchini, G. Prencipe, N. Santoro (Eds.), Distributed Computing by Mobile Entities, Current Research in Moving and Computing, in: LNCS, vol. 11340, Springer, 2019, pp. 615–681.
[8] J.J. Daymude, A.W. Richa, C. Scheideler, The canonical amoebot model: algorithms and concurrency control, in: 35th International Symposium on Distributed Computing, DISC 2021, in: LIPIcs, vol. 209, 2021, pp. 20:1–20:19.
[9] G. D'Angelo, M. D'Emidio, S. Das, A. Navarra, G. Prencipe, Asynchronous silent programmable matter achieves leader election and compaction, IEEE Access 8 (2020) 207619–207634.
[10] A. Navarra, F. Piselli, Asynchronous silent programmable matter: line formation, in: Proc. 25th Int.'l Symp. on Stabilization, Safety, and Security of Distributed Systems (SSS), in: LNCS, vol. 14310, 2023, pp. 598–612.
[11] A. Navarra, F. Piselli, Silent programmable matter: coating, in: Proc. 27th Int.'l Conf. on Principles of Distributed Systems (OPODIS), in: LIPIcs, vol. 286, 2023, pp. 25:1–25:17.
[12] A. Navarra, F. Piselli, Brief announcement: line formation in silent programmable matter, in: Proc. 37th Int.'l Symp. on Distributed Computing (DISC), in: LIPIcs, vol. 281, 2023, pp. 45:1–45:8.
[13] A. Navarra, G. Prencipe, S. Bonini, M. Tracolli, Scattering with programmable matter, in: Proceedings of 37th International Conf. on Advanced Information Networking and Applications (AINA), Advances in Intelligent Systems and Computing, Springer, 2023, pp. 236–247.
[14] Y. Kim, Y. Katayama, K. Wada, Pairbot: a novel model for autonomous mobile robot systems consisting of paired robots, arXiv:2009.14426, 2020.
[15] P. Flocchini, G. Prencipe, N. Santoro, Moving and computing models: robots, in: P. Flocchini, G. Prencipe, N. Santoro (Eds.), Distributed Computing by Mobile Entities, Current Research in Moving and Computing, in: LNCS, vol. 11340, Springer, 2019, pp. 3–14.
[16] I. Suzuki, M. Yamashita, Distributed anonymous mobile robots: formation of geometric patterns, SIAM J. Comput. 28 (4) (1999) 1347–1363.

[17] P. Flocchini, G. Prencipe, N. Santoro, P. Widmayer, Arbitrary pattern formation by asynchronous, anonymous, oblivious robots, Theor. Comput. Sci. 407 (1–3) (2008) 412–447.
[18] M. Yamashita, I. Suzuki, Characterizing geometric patterns formable by oblivious anonymous mobile robots, Theor. Comput. Sci. 411 (26–28) (2010) 2433–2453.
[19] S. Das, P. Flocchini, N. Santoro, M. Yamashita, Forming sequences of geometric patterns with oblivious mobile robots, Distrib. Comput. 28 (2) (2015) 131–145.
[20] N. Fujinaga, Y. Yamauchi, H. Ono, S. Kijima, M. Yamashita, Pattern formation by oblivious asynchronous mobile robots, SIAM J. Comput. 44 (3) (2015) 740–785.
[21] S. Cicerone, G. Di Stefano, A. Navarra, Embedded pattern formation by asynchronous robots without chirality, Distrib. Comput. 32 (4) (2019) 291–315.
[22] S. Cicerone, G. Di Stefano, A. Navarra, Asynchronous arbitrary pattern formation: the effects of a rigorous approach, Distrib. Comput. 32 (2) (2019) 91–132.
[23] S. Cicerone, G. Di Stefano, A. Navarra, Solving the pattern formation by mobile robots with chirality, IEEE Access 9 (2021) 88177–88204, https://doi.org/10.1109/ACCESS.2021.3089081.
[24] S. Cicerone, A. Di Fonso, G. Di Stefano, A. Navarra, Arbitrary pattern formation on infinite regular tessellation graphs, Theor. Comput. Sci. 942 (2023) 1–20, https://doi.org/10.1016/j.tcs.2022.11.021.
[25] G. Prencipe, Pattern formation, in: P. Flocchini, G. Prencipe, N. Santoro (Eds.), Distributed Computing by Mobile Entities, Current Research in Moving and Computing, in: LNCS, vol. 11340, Springer, 2019, pp. 37–62.
[26] S.W. Golomb, D.A. Klarner, Polyominoes, in: Handbook of Discrete and Computational Geometry, 2nd ed., Chapman and Hall/CRC, 2004, pp. 331–352.
[27] S. Cicerone, A. Di Fonso, G. Di Stefano, A. Navarra, MOBLOT: molecular oblivious robots, in: F. Dignum, A. Lomuscio, U. Endriss, A. Nowé (Eds.), AAMAS '21: 20th International Conference on Autonomous Agents and Multiagent Systems, Virtual Event, United Kingdom, May 3-7, 2021, ACM, 2021, pp. 350–358.
[28] Z. Liu, Y. Yamauchi, S. Kijima, M. Yamashita, Team assembling problem for asynchronous heterogeneous mobile robots, Theor. Comput. Sci. 721 (2018) 27–41, https://doi.org/10.1016/j.tcs.2018.01.009.
[29] D. Woods, D. Doty, C. Myhrvold, J. Hui, F. Zhou, P. Yin, E. Winfree, Diverse and robust molecular algorithms using reprogrammable DNA self-assembly, Nature 567 (7748) (2019) 366–372, https://doi.org/10.1038/s41586-019-1014-9.
[30] S. Cicerone, G. Di Stefano, A. Navarra, "Semi-asynchronous": a new scheduler in distributed computing, IEEE Access 9 (2021) 41540–41557, https://doi.org/10.1109/ACCESS.2021.3064880.
[31] S. Cicerone, G. Di Stefano, A. Navarra, A structured methodology for designing distributed algorithms for mobile entities, Inf. Sci. 574 (2021) 111–132, https://doi.org/10.1016/j.ins.2021.05.043.