Swarm Intelligence (2019) 13:59–94
https://doi.org/10.1007/s11721-019-00163-0

Provable self-organizing pattern formation by a swarm of robots with limited knowledge

Mario Coppola[1,2] · **Jian Guo**[2] · **Eberhard Gill**[2] · **Guido C. H. E. de Croon**[1]

Received: 1 May 2018 / Accepted: 28 January 2019 / Published online: 11 February 2019

Abstract
In this paper we present a procedure to automatically design and verify the local behavior of robots with highly limited cognition. All robots are: anonymous, homogeneous, non-communicating, memoryless, reactive, do not know their global position, do not have global state information, and operate by a local clock. They only know: (1) the relative location of their neighbors within a short range and (2) a common direction (North). We have developed a procedure to generate a local behavior that allows the robots to self-organize into a desired global pattern despite their individual limitations. This is done while also avoiding collisions and keeping the coherence of the swarm at all times. The generated local behavior is a probabilistic local state-action map. The robots follow this stochastic policy to select an action based on their current perception of their neighborhood (i.e., their local state). It is this stochasticity, in fact, that allows the global pattern to eventually emerge. For a generated local behavior, we present a formal proof procedure to verify whether the desired pattern will always eventually emerge from the local actions of the agents. The novelty of the proof procedure is that it is primarily local in nature and focuses on the local states of the robots and the global implications of their local actions. A local approach is of interest to reduce the computational effort as much as possible when verifying the emergence of larger patterns. Finally, we show how the behavior could be implemented on real robots and investigate this with extensive simulations on a realistic robot model. To the best of our knowledge, no other solutions exist for robots with such limited cognition to achieve this level of coordination with proof that the desired global property will emerge.

Keywords Pattern formation · Emergence · Self-organization · Formal verification · Liveness · Safety · Robot · Swarm

1 Introduction

The objective of swarm robotics is to enable several robots to collaborate toward a common goal. The goal of pattern formation, which is when the swarm must form a desired spatial configuration, has been a topic of significant attention with many applications for aerial

✉ Mario Coppola
m.coppola@tudelft.nl

Extended author information available on the last page of the article

robots (Achtelik et al. 2012; Saska et al. 2016), underwater robots (Joordens and Jamshidi 2010), satellites (Engelen et al. 2011; Verhoeven et al. 2011), and more. For safety reasons, the behavior should also ensure that collision paths are avoided and that the swarm remains coherent (i.e., the swarm does not break apart into multiple groups). Our principal interest lies in developing a simple behavior to achieve pattern formation with a swarm of robots with extremely low levels of cognition.

One relevant example of an extremely limited robot is miniature quad-rotors, henceforth referred to as micro air vehicles (MAVs). They are characterized by low memory and processing capabilities due to their increasingly small size and mass (McGuire et al. 2016). When operating in closed environments, where Global Navigation Satellite Systems (GNSSs) may be unavailable, they should coordinate only using the relative position of their neighbors, of which they may also be unable to discern the identity, as for instance in the system studied by Faigl et al. (2013) or by Stegagno et al. (2016). Furthermore, intra-swarm communication may prove itself challenging to achieve in practice and is best kept at a minimum (Hamann 2018). For example, our recent experiments showed how a small group of three MAVs can already begin to suffer from relatively limited rate of communication and growing interference (Coppola et al. 2018; van der Helm et al. 2018). Finally, in our pursuit of a minimalist swarm, we also expect all MAVs to be functionally homogeneous without pre-allocated tasks. Mesbahi and Egerstedt (2010) refer to this as "assignment-free." Accepting all these limitations leads us to robots that have no knowledge of their surroundings except (in what we assume to be a minimal requirement for collaboration) the current relative location of their closest neighbors. The motivation behind this work was thus to determine a local behavior with which a swarm of robots with such minimal knowledge could nevertheless be able to both handle safety critical goals (i.e., collision avoidance and swarm coherence) as well as systematically self-organize into a pattern. Moreover, we aimed for a simple reactive behavior that could be concisely stored and processed even by the least capable of robots.

There are two fundamental challenges in the development of swarm behavior for such limited robots:

1. the top-down automatic development of local rules from a global goal,
2. the bottom-up verification of whether the local rules will lead to the desired global goal.

The two main contributions in this paper directly address these two challenges. For our very limited robots, we automatically define the local rules that they must follow in order to form a pattern. As it will be seen, these rules are presented as a probabilistic state-action map that can be automatically generated with a few steps. This is the first main contribution. We then provide a method to automatically verify whether the swarm will always eventually form the pattern, or whether certain other spurious results may occur. The proof procedure has the novel aspect that it focuses on the analysis of local states of the agents, rather than all global states of the swarm, in order to determine the successful formation of the global desired pattern from any other initial pattern. This allows for computation tractability and constitutes the second main contribution.

The generated local behavior of the robots is defined by a probabilistic local state-action map. The local state of a robot is simply a discretized view of its current neighborhood, and the actions are directions that it can move toward. This local state-action map can easily be developed to simultaneously handle collision avoidance, avoidance of swarm separation, and formation of a desired pattern. The swarm acts entirely stochastically only based on this. All robots have the same state-action map. As the robots operate using local clocks, any robot can move at any time. When it does, it uses the probabilistic state-action map to stochastically select its next action out of the available options (with equal probability). The global pattern

emerges from this stochastic process once all robots find themselves in local states in which they cannot select any action to move anymore. This stochastic behavior means that the same pattern will be formed in several different ways even when starting from the same initial conditions, and how the pattern is formed is left to the robots. However, although it may not necessarily be important *how* the goal is reached, it is important that *it is* reached. This is the reason why we present an automatic verification procedure to verify whether the local behaviors will always eventually lead to the intended higher-level behavior.

This paper is organized as follows. We first define the problem in Sect. 2. In Sect. 3, we review other solutions to pattern formation and we explain the context and novelty of our contributions. The methodology is then detailed in Sect. 4. Here, we explain how to generate the probabilistic state-action map and we present the proof procedure to check whether the desired pattern will always eventually emerge. We then perform extensive simulations of an increasing level of fidelity. In this way, we explore different aspects of the behavior, from the more fundamental to the more practical. Specifically, we start with an idealized system operating on a discrete grid in discrete time steps (Sect. 5), moving on to accelerated particles in continuous space, and finally to simulated MAVs with a realistic quad-rotor model and sensor noise (Sect. 6). The insights gathered are further discussed in Sect. 7. Finally, Sect. 8 provides concluding remarks and summarizes future research directions.

2 Problem definition, constraints, and assumptions

The problem tackled in this paper is for a swarm of robots to reshuffle into a pattern while avoiding collisions and group separation. In this work, a pattern P is an anonymous spatial configuration of robots on a 2D plane with specific relative positions to one another.[1] Let P_{des} be the desired final pattern that the swarm settles in. Considering our interest in robotics, P_{des} must be achieved while also avoiding collision paths and swarm separation. More formally, we are interested in achieving a behavior that can ensure that the swarm is **safe** (Definition 1) and **live** (Definition 2).

Definition 1 The swarm is **safe** if neither of the following events occurs: 1) a collision between two or more robots, 2) the swarm disconnects into two or more groups.

Definition 2 The swarm is **live** if, starting from any initial pattern $P_0 \neq P_{des}$, it will always eventually form the desired pattern P_{des}, where the only restriction on P_0 and P_{des} is that they have a connected sensing topology.

The robots have the following constraints:

C1 The robots are homogeneous (all entirely identical).
C2 The robots are anonymous (they cannot sense each other's identity).
C3 The robots are reactive (they only select an action based on their current state).
C4 The robots are memoryless (they do not remember past states).
C5 No robot can be a leader or seed.
C6 The robots cannot communicate with each other.
C7 The robots only have access to their local state.
C8 The robots do not know their global position.
C9 The robots exist in an unbounded space.

[1] This definition of pattern is adapted from the definition used in the context of cellular automata by Sapin (2010).

C10 Each robot can only sense the relative location of its neighbors up to a short range.

The following assumptions are made:

A1 The robots all have knowledge of a common direction (i.e., North).
A2 The robots operate on a 2D plane.
A3 When a robot senses the relative location of a neighbor, it can sense it with enough accuracy and update frequency to establish if a neighbor is moving or standing still (e.g., hovering).
A4 P_0, the initial pattern formed by the robots, has a connected sensing topology.

The rationale behind each assumption is:

- Assumption A1 is a typical assumption in several swarm designs (Ji and Egerstedt 2007; Shiell and Vardy 2016). On real robots, a common direction can be known using on-board sensors such as, but not limited to, a magnetic sensor and/or a gyroscope (Conroy et al. 2005; Oh et al. 2015).
- Assumption A2 is representative of ground robots or MAVs flying at approximately the same height.
- Assumption A3 deserves a more in-depth analysis. For general robotic platforms, relative localization is deemed a fundamental tool for collision avoidance and coordination. Concerning MAVs, for instance, a sufficiently accurate relative localization technology is required if collision avoidance (a basic behavior needed for them to swarm safely) is required. There exist several technologies to achieve relative localization. Pugh et al. (2009) and Roberts et al. (2012) used technology based on infrared (IR) signals. Basiri et al. (2014) introduced an audio-based solution with a microphone array. Faigl et al. (2013) and Roelofsen et al. (2015) proposed vision-based methods relying solely on (one or more) on-board cameras. Coppola et al. (2018) and Guo et al. (2017) explored relative localization sensors based on signal ranging. In this work, we will show that fulfilling Assumption A3 up to a certain extent is paramount to provide safe behavior in spite of all other constraints. In our final simulations, to be found in Sect. 6.3, we will show that in practice the swarm can also function even when the robots are only able to detect movements beyond a certain threshold velocity, rather than if adhering perfectly to the assumption.
- Assumption A4 is needed for the entire swarm to begin acting as a collective. If Assumption A4 were violated (and, for instance, the swarm was to begin as two separate groups that cannot sense each other), then it could not ever be expected for the separate groups to find each other in an unbounded space.

3 Related works and research context

Pattern formation is a well-studied problem in robotics. A review of existing solutions is presented in Sect. 3.1. The swarm treated in this work sets itself apart by its minimalist nature, constraining the knowledge of the robots to only the relative location of nearby neighbors and the North direction. We then discuss the contributions and their context in Sect. 3.2.

3.1 Review of approaches to pattern formation by a swarm of robots

The solutions to pattern formation found in the literature rightfully vary depending on the sensing capabilities of the robots. In this section we review solutions present in the literature, starting from cases where the robots are more knowledgeable of their surroundings to increasingly more minimalist cases more similar to our own (as introduced in Sect. 2).

Several solutions are based on the assumption that each robot in the swarm can directly sense every other robot. In this case, the topology of the swarm is said to be *fully connected* or *complete*. This endows each robot with a global view of the swarm. This type of swarm is found to self-stabilize to an equilibrium only by means of attraction and repulsion forces (Gazi and Passino 2004). Izzo and Pettazzi (2005, 2007) showed how the attraction and repulsion forces alone could be tuned such that the swarm stabilizes into a desired pattern. However, the results had two limitations: (1) the swarm can unpredictably form spurious patterns depending on the initial conditions due to the presence of spurious equilibria, (2) they were limited to symmetric patterns. Asymmetry is difficult for a homogeneous non-communicating swarm to resolve, and it was tackled with the use of neural networks in later work (Izzo et al. 2014; Scheper and de Croon 2016). Formation control algorithms have also been proposed, whereby the robots are allocated positions/distances to achieve and maintain with respect to the other robots (Pereira and Hsu 2008; de Marina Peinado 2016). With this strategy, the swarm will quickly form the desired pattern. However, it is required to specify the necessary inter-robot distances/locations without anonymity.

To address that the swarm may not always begin in a fully connected topology, Ji and Egerstedt (2007) and Mesbahi and Egerstedt (2010) proposed the use of a gathering algorithm so that all robots come together prior to initiating the pattern formation task. In several scenarios, however, being in a *fully* connected topology is simply not viable, and we must accept that the topology of the system is just connected, and not fully connected. For instance, if robots sense each other using on-board cameras or IR sensors, as could likely be the case for MAVs or ground robots, they will be unable to see behind other robots or beyond a certain distance.[2] Tanner (2004) and Rahmani et al. (2009) showed how to control swarms with a static connected topology, yet when the robots can only sense their closest neighbors, the topology of the swarm will not be static, but it will change depending on the current relative positions. Falconi et al. (2010) showed how to combine local positioning information together with a communication protocol in a consensus algorithm. Similarly to formation control, however, this algorithm requires specifying the formation parameters without anonymity. Another popular solution found in the literature is to use seed robots: these are robots in the swarm that do not move and act as a reference to the other robots. Rubenstein et al. (2014) used this to enable an impressively large swarm of simple robots to form shapes. Four seed robots were manually placed in a cross-formation, and the other robots then circled around them and "filled up" the shape. Instead, Wessnitzer et al. (2001) used seed robots to build up patterns in a chain-like fashion, starting from a seed robot that recruits other robots. A seed was also used for a system of self-arranging blocks by Grushin and Reggia (2008, 2010). Here, a static seed block acted as a reference for others to determine their correct relative position (through communication with neighbors), virtually providing them with a global reference albeit while still only making use of local communication. Bonabeau et al. (2000)

[2] As already mentioned near the end of Sect. 2, there is a vast amount of solutions for relative localization in swarm robotics, and it is also a separate topic of exploration in our own current research (Coppola et al. 2018; van der Helm et al. 2018). Here, we declare the challenge outside of the scope of this work and we deem it sufficient to assume that the robots are endowed with the necessary sensors to sense neighboring robots within a short omni-directional range.

also studied the rules for the construction of a structure by robots. The robots would begin by placing blocks next to a seed block according to specific rule sets, whereby the blocks could no longer be moved once a robot had placed them. This created a slowly evolving construction. More recently, Werfel and Nagpal (2008) and Werfel et al. (2014) developed and implemented an algorithm in order to coordinate the construction task for a team of robots. This algorithm also relied on the use of a seed block, which the robots could use as a unique shared reference to determine where to place the other blocks. However, in general, the use of a reference (which for pattern formation would be a seed robot) requires that other robots can identify it, which is not the case here given that the robots are all anonymous. Moreover, when they are all functionally homogeneous, no robot can be assigned as the seed. Without communication, they cannot elect one themselves either, as otherwise explored by Yamauchi and Yamashita (2014), Derakhshandeh et al. (2016), and Di Luna et al. (2017), where a swarm could self-elect a leader/seed robot.

We now move to even simpler systems. For homogeneous and anonymous robots with no seeds, Klavins (2002) proposed to encode a pattern as a graph and a collection of its sub-graphs. This technique set the way for the use of graph grammars, later developed in Klavins (2007) for self-assembly by a team of robots. The robots randomly drifted in a confined environment and could latch together upon encounter. Once latched, they could communicate their state and determine whether the connection formed a part of the total graph, in which case they would remain attached. Otherwise, they would detach and continue drifting. Using this approach, the pattern would slowly assemble. Similar strategies were studied by Smith et al. (2009), Arbuckle and Requicha (2010), Arbuckle and Requicha (2012), Fox and Shamma (2015). In more recent work, Haghighat and Martinoli (2017) proposed an algorithm for the automatic encoding of such rules for rotationally symmetric modules. However, the local rules used in these studies do not incorporate the additional fundamental constraints of the robots that are studied in this work, namely that the robots cannot: collide, latch together, randomly drift apart, or (most importantly for these algorithms to work) communicate. Without communication it is not possible for the assembly to grow, because the robots are not capable of knowing more than their local state at any point and thus require a different decision-making process on the level of the individual agent.

Intra-swarm communication is a very powerful tool. It allows robots to share their intentions and their perspectives. It was used in several works that we already discussed and more, including consensus algorithms (Falconi et al. 2010, 2011, 2015), leader-election algorithms (Di Luna et al. 2017), or bidding algorithms for task allocation (Gerkey and Matarić 2004). More recently, Slavkov et al. (2018) studied how to use a communication architecture to diffuse activation values across the swarm. The swarm could then rearrange itself so as to protrude in regions of high activation values, creating emergent morphologies. Communication can also double as a sensor. Nembrini et al. (2002) and Winfield et al. (2008) used communication to enable a swarm to remain connected even in the presence of obstacles by repeatedly checking for connectivity with the neighbors through a broadcast and listening protocol. In Winfield and Nembrini (2012), the robots communicate their adjacency matrix to one another in order to extend their knowledge beyond what their sensors allow, which is found to increase the coherence performance.

Despite its advantages, considering the practical difficulties in ensuring a high-throughput and reliable intra-swarm wireless communication (Hamann 2018; Coppola et al. 2018), we have taken an interest in establishing a behavior that also does not natively require communication, such that it can work even when such hardware is not available. Once even communication is removed, few works, to the best of our knowledge, explore the coordination of a swarm of robots that is as limited as the one presented in this work. Krishnanand and

Ghose (2005) developed alignment behaviors by which they could form non-finite grids and lines. Flocchini et al. (2005) explored the gathering problem, whereby all robots must aggregate together as much as possible. Yamauchi and Yamashita (2013) examined the formation power of very limited agents, but a behavior to achieve the patterns was not developed. This leaves a knowledge gap in the field of minimalist swarming.

3.2 Contributions and research context

There are two principal scientific contributions in this paper:

1. An automatic procedure to extract the local behavior so that a swarm of robots with extremely limited cognition and no communication can form a desired pattern, while also avoiding collisions and maintaining a connected sensing topology.
2. An automatic proof procedure to verify whether the set of local rules will always eventually cause the swarm to generate the pattern. We present a primarily *local* analysis of the behavior which allows to verify that the *global* pattern can be achieved from any initial pattern P_0. The large advantage of such a local analysis is that it limits the computational explosion of global proof methods.

Automatic procedures to generate local rules that create high-level functions are already present in the literature. Two notable recent works in this domain are from Rubenstein et al. (2014) and Werfel et al. (2014). Both systems demonstrate an efficient distributed behavior. Looked at from above, we see that the global goal is slowly reached by the robots. The difference with our work stems from the limitations of our robots, which do not (and cannot, in light of their limited cognition) rely on a reference. As a result, their behavior is fully dictated by their local environment without any global context. Furthermore, unlike the system tackled by Grushin and Reggia (2008, 2010), our robots also cannot see far, meaning that they do not know what they will find when they move. Therefore, they cannot knowingly move toward local target locations. This is why they must rely on a probabilistic scheme.

The final pattern is automatically encoded from the larger pattern within the state-action map under this rule: if a robot finds itself in a local state that *may* constitute the global desired pattern, it will not take any action. This eventually gives rise to the pattern once all robots end up in such states. Conceptually, the breakdown of a large pattern into smaller parts resembles graph grammar approaches, as for instance used by Klavins (2007) or Haghighat and Martinoli (2017). In our case, however, the robots cannot communicate and must only use the knowledge that a neighbor is (or is not) there in order to decide their next action. Furthermore, the robots cannot detach and drift freely, which restricts how the swarm can evolve. Overall, this means that the pattern does not slowly assemble, but rather forms by the stochastic (inter)actions of the robots. The phenomenon can only be detected at the macroscale and not by the robots themselves. This behavior is characteristic to emergent processes (Bonabeau and Dessalles 1997), and its complexity is the reason that we also need to verify that our desired pattern is the sole emergent result.

Our verification of the emergent property (i.e., the final pattern) is based on a formal analysis of the swarm, inspired by Winfield et al. (2005). Dixon et al. (2012) and Gjondrekaj et al. (2012) applied this with the use of model checking and demonstrated its potential. However, an issue with model checking is that it performs an exhaustive search of all global states (Clarke et al. 1999) and it is subject to a computational explosion as the size of the swarm grows. Konur et al. (2012) tackled this using macroscopic swarm models. These models efficiently describe the evolution of the swarm by means of one finite state machine (Winfield et al. 2008). However, macroscopic models typically assume that robots are uniformly

distributed, or, in general, make probabilistic assumptions about the presence of robots in a given area (Lerman et al. 2001; Prorok et al. 2011). These assumptions may be suitable for more abstract spatial goals, such as aggregation, exploration, or coherence, but they do not apply to pattern formation, which by definition has a strict requirement on the spatial arrangement. To be able to verify the emergent property yet keep the computations low, we focus on a local analysis of the behavior. With this novel analysis, we provide a set of local conditions that, if met, guarantee that the swarm will always eventually self-organize into the desired pattern. Unlike the macroscopic models discussed above, this analysis means that we do not merely *assume* that there is enough free movement/motion in the swarm, but use the conditions to check that this is in fact the case. With this, we limit the global analysis only to the discovery of spurious patterns. However, this search only needs to be executed on a very restricted subspace, for which we provide a methodology to identify the candidates.

4 Designing and verifying the behavior of the robots

This section describes the design and verification of the probabilistic local state-action map that dictates the behavior of the robots. We detail how the state-action map can be crafted such that the swarm will remain safe (Definition 1) and (possibly) also live (Definition 2). As we are dealing with robots with extremely limited knowledge, it can be expected that it is not always the case that both properties can be achieved at the same time. Safety is a hard requirement, but it will naturally restrict the ways in which the swarm can evolve. This could lead the swarm to a **livelock**.

Definition 3 A **livelock** is a situation in which the swarm will endlessly transition through a set of patterns (e.g., $P_0 \rightarrow P_1 \rightarrow P_2 \rightarrow P_0 \rightarrow P_1 \rightarrow P_2 \rightarrow P_0 \ldots$) and cannot transition to any other patterns.

Furthermore, the limited view that the robots have of their surroundings limits the knowledge that they have of the structure, which may cause other (perhaps undesired) patterns to form. We will refer to this situation as **deadlock**.

Definition 4 A **deadlock** is a situation in which the swarm forms an undesired pattern $P \neq P_{des}$, where no robot in the swarm can take action.

We have developed proof procedures to verify that livelocks or deadlocks will not happen. We will provide a set of conditions and checks that, if fulfilled, guarantee that the state-action map constructed for a given pattern is such that livelocks and deadlocks do not occur, and thus imply that the swarm is safe *and* live. The state-action map is developed and verified in a formal domain, assuming robots to be idealized agents existing on a 2D grid and operating in discrete time. Although this may seem restrictive, we will show in Sect. 6 how it can be used on robots operating in a realistic setting. The idealized framework is described in Sect. 4.1, and the method to design the probabilistic state-action map is detailed in Sect. 4.2. The conditions to prove whether a state-action map is safe, free of livelocks, and free of deadlocks are provided in Sects. 4.3, 4.4, and 4.5, respectively.

4.1 The formalized framework

Consider N agents (idealized robots) that exist in an unbounded discrete 2D grid. Each robot is endowed with short-range omni-directional relative sensors and knowledge of North. In

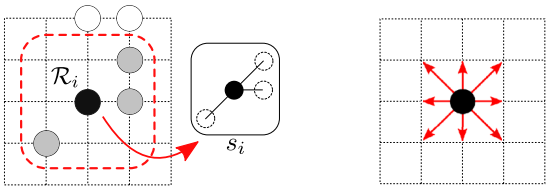


Fig. 1 Depictions of local state and the actions that an agent can take as used in this paper

(a) Example of an agent (black circle) in a local state s_i, given by the relative positions of its neighbors (white circles)

(b) Possible actions that an agent can take. It can move omnidirectionally on the grid

this paper we will focus our attention to robots with omni-directional sensing and motion capabilities, albeit the concepts presented hold for other state spaces and action spaces as well.

In the idealized case, each agent $\mathcal{R}_i$ can sense the location of its neighbors in the 8 grid points that surround it (Fig. 1a). Let s_i be the current state of agent $\mathcal{R}_i$, and let $\mathcal{S}$ be the local state space of the agents. It follows that $|S| = 2^8$, as it represents all local combinations of neighbors that could be sensed. To represent omni-directional motion, the agents are also able to move to any of the 8 grid points surrounding it, as depicted in Fig. 1b. This forms the action space of the agents, denoted $\mathcal{A}$. Note that other discretizations of $\mathcal{S}$ or $\mathcal{A}$ could also apply depending on the sensors and motors available on the robot of interest.

At time step $k = 0$, we assume the swarm begins in an arbitrary pattern P_0 on the grid. The only restriction on P_0 is that it has a connected sensing topology (Assumption A4). At each discrete time step, a random agent in the swarm takes an action and moves to a new location on the grid.[3]

4.2 Developing the probabilistic state-action map

In analogy to biological systems, the behavior that we will design replicates these three rules:

1. *be careful* (do not take actions that are in collision course with others);
2. *be social* (do not take actions whereby the swarm might locally break apart);
3. *be happy* (when in a desired local state, do not move).

Let us begin with the full state-action map, given by $\Pi = \mathcal{S} \times \mathcal{A}$. With Π, any agent $\mathcal{R}_i$ in any state $s_i \in \mathcal{S}$ can stochastically take any action in $\mathcal{A}$. Naturally, this can readily cause both collisions and/or group separation, which we want to avoid (if the swarm separates, then there is a chance that the two groups will never find each other, since they are operating in an unbounded environment). Therefore, we scan through Π to identify all state-action pairs that:

(a) *are in the direction of a neighbor*
These state-action pairs will lead to collisions (two agents occupying the same grid point). They form the set $\Pi_{collision}$.

[3] At first sight, this seems rigid and difficult to implement on real robots. It can be in part justified under the intuition that the probability that two robots with different internal clocks begin to move at *exactly* the same time is small. A similar assumption was also suggested by Winfield et al. (2005) as a method to model random concurrency in the swarm. In Sect. 6 we will show that, if robots are able to sense whether their neighbors are taking an action (assumption A3 from Sect. 2), then it can be exported to real robots. Multiple robots within the swarm will move, yet locally only one neighbor will move on a first-come first-served basis. In the idealized system, this is simplified to only one robot moving at one time step.

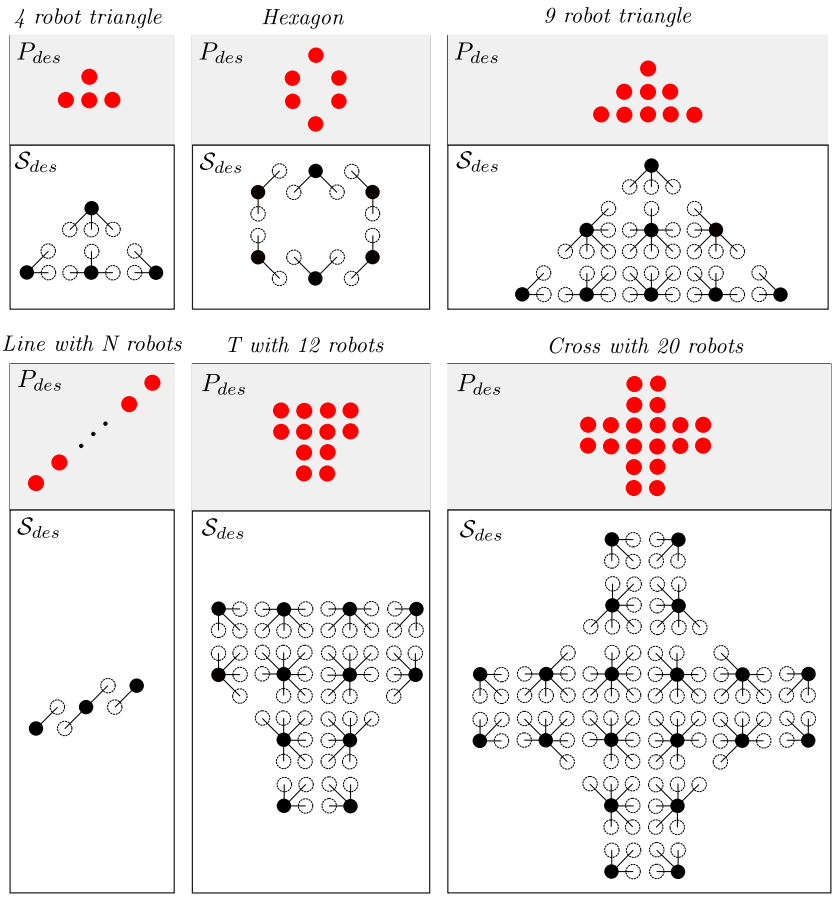


Fig. 2 Examples of patterns and their respective desired states $\mathcal{S}_{des}$. The set $\mathcal{S}_{des}$ can be intuitively extracted from a pattern P_{des}, making it easy for a designer to define the local behavior of the robots

(b) *may cause the swarm to become disconnected*
These actions will break the local connectivity of the agents (the local neighborhood splits into two or more groups). They form the set $\Pi_{separation}$.

We then define Π_{safe}:

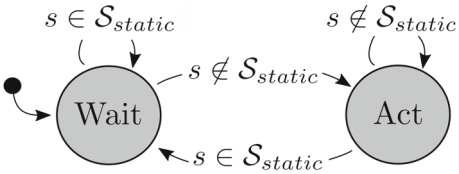
$$\Pi_{safe} = \Pi - (\Pi_{collision} \cup \Pi_{separation}). \tag{1}$$

If the agents follow Π_{safe}, we can guarantee that the swarm remains safe while randomly reshuffling. The proofs for this are provided in Sect. 4.3.

Π_{safe} can be further modified to also make a desired pattern form. To do this, let us extract the set of local states that the agents are in when the desired pattern P_{des} is achieved. This forms a set of local *desired* states, denoted $\mathcal{S}_{des}$, examples of which are shown in Fig. 2 for different patterns.

If an agent $\mathcal{R}_i$ finds itself in a state $s_i \in \mathcal{S}_{des}$, then it should not move. The rationale behind this is that, from its perspective, the goal has been achieved (although this may or may not be the case at the global level, the robot does not know this). Therefore, for these states, we exclude all possible actions. The state-action map to form a given pattern P_{des} is:

Fig. 3 FSM of agent behavior



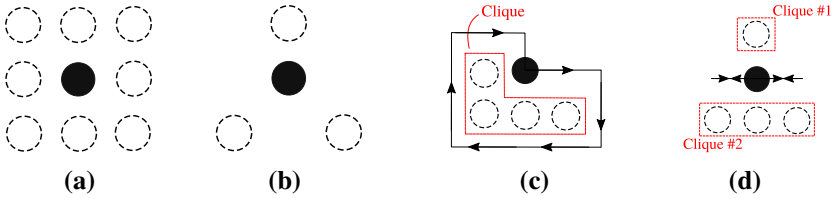


Fig. 4 Examples of an agent (black circle) in different states depending on the relative positions of its neighbors (white circles). Specifically, **a** a state $s \in \mathcal{S}_{blocked}$, all actions will cause a collision; **b** a state $s \in \mathcal{S}_{blocked}$, all actions will either cause a collision or the local topology to disconnect; **c** a state $s \in \mathcal{S}_{active} \cap \mathcal{S}_{simplicial}$, its neighbors form one clique, which allows it to (potentially) travel freely away from or around its neighborhood; **d** a state $s \in \mathcal{S}_{active} \cap \mathcal{S}_{\neg simplicial}$, its neighbors form two cliques, the agent can move, but it cannot leave its neighborhood

$$\Pi_f = \Pi_{safe} - (\mathcal{S}_{des} \times \mathcal{A}). \tag{2}$$

With Π_f, the robots are capable of moving around until the swarm self-organizes into the desired pattern. Sections 4.4 and 4.5 provide the procedures to prove whether Π_f is such that the desired pattern always eventually forms from any initial pattern P_0.

The states in $\mathcal{S}$ can be divided into three groups:

Desired When in these states, the agent should not move. Π_f does not map these states to any action. Desired states are grouped in the set $\mathcal{S}_{des}$.

Blocked These are all states in $\mathcal{S} - \mathcal{S}_{des}$ where the agent cannot move because all actions are unsafe. Π_f does not map these states to any action. We group these states in the set $\mathcal{S}_{blocked}$.

Active These are states that Π_f maps to one or more actions in $\mathcal{A}$. We group these states in the set $\mathcal{S}_{active}$.

Functionally speaking, $\mathcal{S}_{blocked}$ and $\mathcal{S}_{des}$ are equivalent. In either case, the agent will not move. Based on this, we also define the superset $\mathcal{S}_{static} = \mathcal{S}_{des} \cup \mathcal{S}_{blocked}$. Overall, the local behavior of an agent is summarized by the finite state machine (FSM) in Fig. 3. Two examples of blocked states are shown in Fig. 4a, b.

Additionally to the taxonomy above, we also define a set of states as **simplicial**.[4]

Definition 5 A **simplicial** state is a state $s \in \mathcal{S} - \mathcal{S}_{blocked}$ for which its neighbors form only one clique.

Definition 6 A **clique** is a connected set of an agent's neighbors.

Simplicial states are grouped under the set $\mathcal{S}_{simplicial}$. All states in $\mathcal{S}$ that are not simplicial are denoted $\mathcal{S}_{\neg simplicial}$. From this, it follows that $\mathcal{S}_{blocked} \subseteq \mathcal{S}_{\neg simplicial}$. An example of

[4] These definitions are borrowed from, but not equivalent to, the typical definitions of *simplicial node* and *clique* (Van Steen 2010). In standard graph theory, a simplicial node is a node whose neighboring nodes are *fully* connected among each other, not just connected. Similarly, a clique is a fully connected set of neighbors, whereas in our case it is just a connected set.

a state that is both simplicial and active is shown in Fig. 4c. By contrast, a non-simplicial active state is shown in Fig. 4d. An agent in a simplicial state could potentially move without risking that the swarm ceases to be in a connected topology, unlike the non-simplicial case. Intuitively, agents who happen to be in a simplicial state thus have the potential to travel freely across the swarm and break livelocks. For this reason, simplicial states are going to be an important element to the local proof procedure to determine whether the swarm is free of livelocks, which can be found in Sect. 4.4.

4.3 Verifying safety

Our swarm consists of several agents that can choose to take actions at any point in time. Safety can be guaranteed when agents do not simultaneously perform conflicting actions. To formalize this, we bring forward Proposition 1.

Proposition 1 *If the swarm never features more than one agent moving at the same time, then the swarm can remain safe.*

Proof Consider a connected swarm organized into an arbitrary pattern P. At a given time $t = t_1$, agent $\mathcal{R}_i$ decides to take an action based on action space $\mathcal{A}$. This action should last until $t = t_2$. However, at time $t_1 < t < t_2$, an unsafe event takes place. It follows that the event must have been the fault of agent $\mathcal{R}_i$, because it was the only agent that moved. Therefore, if agent $\mathcal{R}_i$ could select only from safe actions, this would be sufficient to guarantee that the swarm is safe at time $t = t_2$. □

Proposition 1 only applies to the idealized system and cannot be implemented on the real system where robots use local clocks. This explains the importance for Assumption A3 from Sect. 2: an agent must know whether its neighbors are executing an action. If then a robot does not move whenever one of its neighbors is moving (on a first-come-first-served basis), then the swarm can locally approach the formal requirement of Proposition 1 even if several robots may be moving in different neighborhoods. We will return to this in Sect. 6.

Under the assumption that the conditions of Proposition 1, if Π_{safe} meets the conditions in Propositions 2 and 3, then the swarm is safe.

Proposition 2 *If an agent is the only agent moving in the entire swarm, and Π_{safe} is such that the agent can only select actions in directions that can be sensed by its on-board sensors, then no collisions will occur in the swarm.*

Proof Consider an agent $\mathcal{R}_i$ in a swarm. Following Proposition 1, we know that the agent will be the only agent to move. The agent moves in the environment according to the action space $\mathcal{A}$. If all actions in $\mathcal{A}$ lead to a location that is already sensed, then agent $\mathcal{R}_i$ can establish whether the action will cause a collision, and it can choose against performing these actions. □

Proposition 3 *If an agent is the only agent moving in the entire swarm, and Π_{safe} is such that the agent can only select actions where, at its next location, all its prior neighbors and itself remain connected, then the whole swarm will remain connected.*

Proof Consider a connected swarm of N agents. The graph of the swarm is connected if any node (agent) $\mathcal{R}_i$ features a path to any other node (agent) $\mathcal{R}_j$. Consider the case where agent $\mathcal{R}_i$ takes an action. If, following the action, agent $\mathcal{R}_i$ is still connected to all its original

neighbors, then the connectivity of the graph was not affected. If agent $\mathcal{R}_i$ only selects actions where, at its final position, this principle is respected, then it will be able to move while guaranteeing that the swarm remains connected. □

4.4 Verifying against the presence of livelocks

We now provide the proof procedure to check that the system can form the patterns and will do so without ending up in livelocks. Let us begin at the global level and define a directed graph $G_P = (V_P, E_P)$. The vertices V_P represent all possible patterns that the swarm could generate. The edges E_P represent all global pattern transitions that could take place whenever one agent in the swarm executes an action from Π_f. Our final objective is to establish whether Π_f is such that G_P always features a path from any vertex (i.e., an arbitrary initial pattern P_0) to the global desired pattern P_{des}. If this is the case, then it is proven that livelocks will not occur.

This problem could be tackled by directly inspecting G_P, but an exhaustive computation of G_P quickly becomes intractable (Dixon et al. 2012). Otherwise, livelocks (if existent) could be found using heuristic search algorithms, as done by Sapin (2010) to find loops (gliders) for Game of Life Cellular Automata. However, should we not find any, then it is not guaranteed that livelocks do not exist. It only means that the heuristic search did not find them. We thus take a different route and extract local conditions that, if respected, *also guarantee the global property*. Although this comes at the cost of imposing certain local restrictions that may not necessarily be required at the global level, it bears the advantage that they can be verified at the local level and thus independently of the number of robots in the swarm.

In the following analysis, it is assumed that P_0 always has a connected sensing topology (Assumption A4) and that it has N_{des} agents, where N_{des} is the number of agents required to form P_{des}. We also assume that deadlocks are not present. This is not required and is merely done for simplicity. The absence of deadlocks can be verified independently by the methodology in Sect. 4.5.

4.4.1 Ensuring motion

We begin by showing that, if no deadlocks are present, then any pattern $P \neq P_{des}$ will always have at least one agent in an active state, as per Lemma 1.

Lemma 1 *For a swarm of N_{des} agents, if $\mathcal{S}_{static}$ is such that the desired pattern P_{des} is unique (i.e., no deadlocks can occur), any arbitrary pattern $P \neq P_{des}$ will feature at least 1 agent with a state $s \in \mathcal{S}_{active}$.*

Proof By definition: $\mathcal{S}_{static} \cap \mathcal{S}_{active} = \emptyset$ and $\mathcal{S}_{static} \cup \mathcal{S}_{active} = \mathcal{S}$. For a swarm of N_{des} agents that can be in states $s \in \mathcal{S}$, N_{des} instances of states $s \in \mathcal{S}_{static}$ can only coexist into P_{des}, which is known to be the unique outcome. Therefore, it follows that any other pattern must feature at least one agent that is in a state $s \notin \mathcal{S}_{static}$, meaning that it is in a state $s \in \mathcal{S}_{active}$. □

Lemma 1 says that if the swarm cannot be in a deadlock, then it must always have at least one agent that is active, unless P_{des} forms. Therefore, if we can establish that no livelocks can occur, then we know that the swarm will always eventually self-organize into P_{des}. To do this, we need to analyze the local state transitions that an agent can experience over time.

4.4.2 The local state transition graphs

To conduct a local analysis, let us look at Π_f and define its role from the perspective of an agent. When an agent in the swarm experiences a transition from state s to a state s', this can be due to three events:

Event 1 The agent was in a state $s \in \mathcal{S}_{active}$ and computed an action in Π_f. When this happens, some neighbors may disappear from view, while new neighbors may come into view.

Event 2 The agent did not move, but one of its neighbors did. In this case, the neighbor may also have moved out of view.

Event 3 The agent did not move, but some other agent which was previously not in view has moved into view and has become a new neighbor.

Based on the above, let $G_{\mathcal{S}} = (V_{\mathcal{S}}, E_{\mathcal{S}})$ be a directed graph where each vertex $V_{\mathcal{S}}$ represents a different local state $s \in \mathcal{S}$, such that $V_{\mathcal{S}} = \mathcal{S}$, and the edges $E_{\mathcal{S}}$ represent all local state transitions that an agent could experience. More specifically, let us define $E_{\mathcal{S}} = E_1 \cup E_2 \cup E_3$, where E_1 are all edges describing Event 1, E_2 are all edges describing Event 2, and E_3 are all edges describing Event 3. Similarly, $G^1_{\mathcal{S}} = (V_{\mathcal{S}}, E_1)$, $G^2_{\mathcal{S}} = (V_{\mathcal{S}}, E_2)$, $G^3_{\mathcal{S}} = (V_{\mathcal{S}}, E_3)$. The graphs $G^1_{\mathcal{S}}$, $G^2_{\mathcal{S}}$, and $G^3_{\mathcal{S}}$ are illustrated in Fig. 5.

4.4.3 Local achievability of desired states

As a prerequisite for a pattern to form, we require that Π_f ensures that any local state can experience a local transition to a desired local state. If this is the case, we will say that the pattern is **achievable**, as defined by Definition 7.

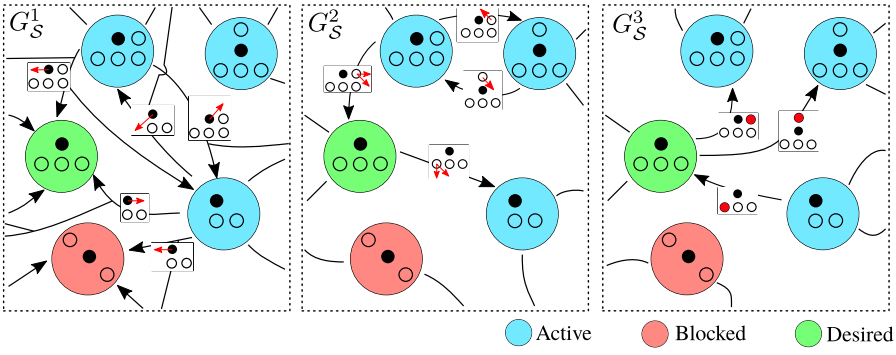


Fig. 5 Exemplary depiction of portions of $G^1_{\mathcal{S}} = (V_{\mathcal{S}}, E_1)$, $G^2_{\mathcal{S}} = (V_{\mathcal{S}}, E_2)$, and $G^3_{\mathcal{S}} = (V_{\mathcal{S}}, E_3)$ (from left to right). Green nodes indicate a desired state, blue nodes indicate an active state, and red nodes indicate a blocked state. The states are visually depicted within each node, showing the agent (in black) and its neighbors. In $G^1_{\mathcal{S}}$ the edges E_1 represent transitions where the agent itself executes an action (shown by the arrows), from which it may probabilistically end up in several local states depending on what it finds after it has moved (notice the bifurcations in the arrows). Note how in $G^1_{\mathcal{S}}$ both the green node (desired) and the red node (blocked) act as sinks, because in these states the agent will not take actions. In $G^2_{\mathcal{S}}$ the edges E_2 represent state transitions experienced by the agent when a neighbor of the agent executes an action. This is shown by one of the neighbors (in white) taking an action. Finally, in $G^3_{\mathcal{S}}$ the edges E_3 represent state transitions that occur when another agent moves into view and becomes a new neighbor. This is shown by the red agents in the transitions

Definition 7 A pattern P_{des} is **achievable** if all local states $\mathcal{S}_{des}$ can be reached starting from any local state in $\mathcal{S}$.

If a pattern is achievable, then there are no restrictions on the local states that can be present in P_0, else there might be certain starting patterns with agents in local states that are unable to transition to certain desired states. This is proven by Lemma 2.

Lemma 2 *If the digraph $G^1_{\mathcal{S}} \cup G^2_{\mathcal{S}}$ shows that each state in $\mathcal{S}$ features a path to each state in $\mathcal{S}_{des}$, then P_{des} is achievable independently of the local states that compose P_0.*

Proof P_{des} is formed if and only if all agents have a state $s \in \mathcal{S}_{des}$, where $\mathcal{S}_{des} \subseteq \mathcal{S}$. Consider an arbitrary initial pattern P_0 for which the local states of the agents form an arbitrary set $\mathcal{S}_0$. Via Lemma 1 we know that there is at least one agent in the swarm that is active for any pattern $P_0 \neq P_{des}$, and in turn any set of states $S_0 \neq S_{des}$. As the active agents move, they will experience transitions described by $G^1_{\mathcal{S}}$, and their neighbors will experience transitions described by $G^2_{\mathcal{S}}$. By the unified graph $G^1_{\mathcal{S}} \cup G^2_{\mathcal{S}}$ we describe the local transitions that an agent experiences as it moves and as its neighbors move. Consider a state $s \in \mathcal{S}_0$ that is incapable (either by its own actions or by the actions of its potential neighbors) to transition to a state in $\mathcal{S}_{des}$. It follows that having this state in S_0 may mean that a state in S_{des} cannot be achieved, and in turn that P_{des} cannot be realized. However, if it is possible for any state in $\mathcal{S}$ to experience local transitions such that it may reach any state $\mathcal{S}_{des}$, it follows that P_{des} is achievable independently of the local states that compose P_0 (i.e., the set $\mathcal{S}_0$), because there is no state $s \in \mathcal{S}_0$ that is incapable of experiencing the necessary transitions that would lead it to be in a state $\mathcal{S}_{des}$. By purposely ignoring the role of $G^3_{\mathcal{S}}$, we restrict the analysis such that:

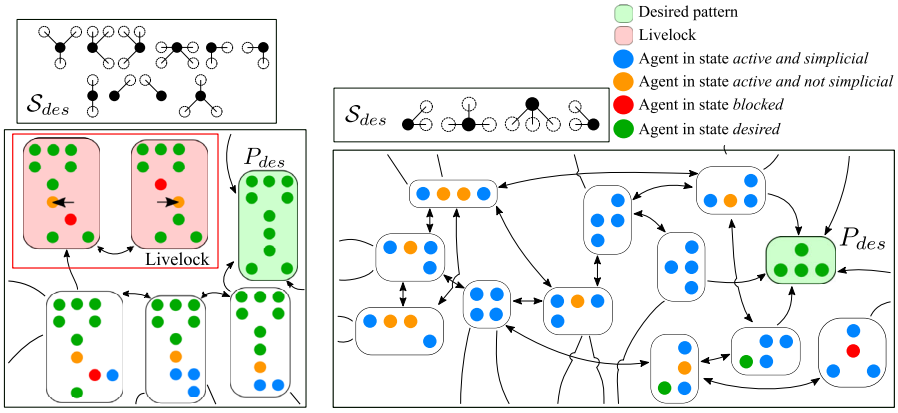
1. Any state s that has too few links for a desired state will have to be active and move to a position where it is surrounded by enough agents. It cannot wait for a local desired state to arise by other agents moving in from outside of its neighborhood.
2. Any state $s \in \mathcal{S}_{blocked}$ can only become active by the actions of a neighbor.
3. The transitions that occur must occur because of changes in the local neighborhood.

This additional restriction ensures that the system can rely on the actions of an agent and/or its neighbors. □

By fulfilling the condition of Lemma 2, we ensure that any initial state could potentially turn into a desired state and avoid placing local-level restrictions on P_0. However, this is still only a local property, and it does not yet fully confirm that, at the global level, P_{des} will always eventually form from any initial pattern P_0, which is the property that we wish to verify. We continue our analysis in Sect. 4.4.4.

4.4.4 Ensuring the presence of agents with simplicial states

In Sect. 4.2 we have already discussed that an agent in a state $s \in \mathcal{S}_{simplicial} \cap \mathcal{S}_{active}$ can potentially move away from its neighborhood. This is an important property. Intuitively, an agent in this state has sufficient freedom for the swarm to escape any livelock. To exemplify this, let us once again consider the global graph G_P as introduced at the beginning of Sect. 4.4, and consider the example in Fig. 6a. When the global pattern formed by the swarm is such that no agent is in a simplicial state, then the swarm is unable to exit the livelock. There is an agent in the swarm that can move, but, because Π_f is designed to keep the swarm safe, it cannot leave its neighborhood and can only move left and right. The result is that the swarm



(a) Example of a livelock **(b)** Pattern where a livelock is not possible

Fig. 6 Illustrations of how a swarm can transition between different patterns, based on movements of the agents that are in active states. More specifically, the figure shows a portion of G_P for two possible desired patterns. The arrows between the nodes are swarm transitions that happen as a result of one of the robots taking an action. Notice that the livelock in **a** does not feature any agents with a state that is both active and simplicial. There is an agent in an active state (in the middle), but because it is not simplicial, it cannot escape its neighborhood and repeatedly moves right and left, causing the livelock

cycles endlessly between the two patterns. By contrast, the patterns in Fig. 6b always have an agent in a simplicial state and no livelocks occur. In this section, we introduce the local conditions necessary such that any vertex (pattern) in G_P always eventually transitions to a pattern with at least one agent with a state that is both active and simplicial (unless P_{des} is reached). This will be an important stepping stone to the final verification in Sect. 4.4.5.

Let P_{AS} be the set of all patterns where one or more agents are in a state $s \in \mathcal{S}_{simplicial} \cap \mathcal{S}_{active}$ (the subscript "AS" stands for "Active and Simplicial"). We wish to ensure a pattern $P \in P_{AS} \cup P_{des}$ will be reached from any other pattern. This is verified via Lemma 3. In this lemma we also make use of a graph $G^{2r}_{\mathcal{S}} \subseteq G^{2}_{\mathcal{S}}$, which only considers the transitions in $G^{2}_{\mathcal{S}}$ that do not feature a neighbor leaving the neighborhood when moving, but only holds transitions about the agent. We also single out a special state in $\mathcal{S}_{blocked}$, which is the one that is fully surrounded by neighbors as in Fig. 4a. We refer to this state as $s_{surrounded}$.

Lemma 3 *If the following conditions are satisfied:*

1. *for all states* $s \in \mathcal{S}_{static} \cap \mathcal{S}_{\neg simplicial} - s_{surrounded}$*, none of the cliques of each state can be formed only by agents that are in a state* $s \in \mathcal{S}_{des} \cap \mathcal{S}_{simplicial}$*,*
2. $G^{2r}_{\mathcal{S}}$ *shows that all static states with two neighbors will directly transition to an active state,*

then a pattern in $P \in P_{AS} \cup P_{des}$ *will always be reached from any other pattern* $P \notin P_{AS} \cup P_{des}$*.*

Proof Consider an agent $\mathcal{R}_i$ with state $s_i \in \mathcal{S}_{static} \cap \mathcal{S}_{\neg simplicial}$. By definition, s_i must have more than one clique, unless $s_i = s_{surrounded}$. If $s_i = s_{surrounded}$ and $P \neq P_{des}$ then one of $\mathcal{R}_i$'s neighbors must be in a state $s \in \mathcal{S}_{active} \cap \mathcal{S}_{simplicial}$, or else there must exist other agents beyond $\mathcal{R}_i$'s direct neighborhood. If $s_i \neq s_{surrounded}$, then the neighbors of agent $\mathcal{R}_i$ form two or more cliques. In all cases, the pattern $P \neq P_{des}$ extends in two or more directions that stem from agent $\mathcal{R}_i$. If we trace any branch, because only a finite number of agents N_{des} exists, we have the two following possible situations:

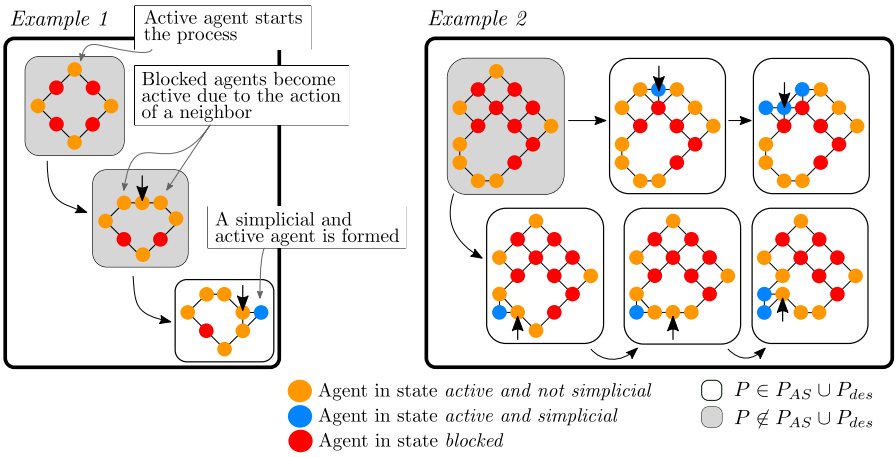


Fig. 7 Illustration of two exemplary loops that "collapse." Notice that the active states present at the borders cause a chain reaction until eventually a simplicial active agent is present. This is a property that can be determined by inspecting $G_{\mathcal{S}}^{2r}$, which will show that the static agents will become active and propel the chain reaction

1. The branch eventually features an agent $\mathcal{R}_j$ with state $s_j \in \mathcal{S}_{simplicial}$. In the extreme, this is a leaf on the edge of the pattern. Here, we can have two situations:
   (a) $s_j \in \mathcal{S}_{des} \cap \mathcal{S}_{simplicial}$. If this exists, then the simplicial agent is also static. Therefore, it is possible that the entire pattern does not feature *any* active and simplicial agent.
   (b) If $s_j \notin \mathcal{S}_{des} \cap \mathcal{S}_{simplicial}$, then $s_j \in \mathcal{S}_{active} \cap \mathcal{S}_{simplicial}$ and so we are done.

   If, by design, states $s \in \mathcal{S}_{des} \cap \mathcal{S}_{simplicial}$ cannot be combined to form the clique of a state in $\mathcal{S}_{static} \cap \mathcal{S}_{\neg simplicial} - s_{surrounded}$, then it is guaranteed that $s_j \notin \mathcal{S}_{des} \cap \mathcal{S}_{simplicial}$. Therefore, we can *locally* impose that situation (b) always occurs, that situation (a) never occurs, and we thus guarantee that $s_j \in \mathcal{S}_{active} \cap \mathcal{S}_{simplicial}$. This is the first condition of this Lemma.
2. If all branches from agent $\mathcal{R}_i$ only feature non-simplicial states, then this is only the case if the branches form loops, otherwise at least one leaf would be present as in situation 1. However, it can be ensured that a loop will always collapse and feature one simplicial active agent. In a loop, all agents have two cliques, each formed by one neighbor. $G_{\mathcal{S}}^{2r}$ tells whether any static agent with two neighbors, by the action of its neighbors, will become active. This is the second condition of this Lemma. If this is the case for all states, then we know that the action of any neighbor will cause a chain reaction about the loop. This will eventually cause the loop to collapse about one corner point and create a simplicial leaf, unless P_{des} forms. In either case, we reach a pattern $P \in P_{AS} \cup P_{des}$. The collapse of two exemplary loops is depicted in Fig. 7.

In summary, by creating the conditions such that situation 1(a) never occurs, we restrict the possible patterns that can exist outside of $P_{AS} \cup P_{des}$ to patterns with only loops (situation 2). If P_0 is a loop, then through $G_{\mathcal{S}}^{2r}$ we know that loop patterns will collapse into a pattern that exists within $P_{AS} \cup P_{des}$. Else, P_0 already exists within $P_{AS} \cup P_{des}$. This means that any pattern P_0 will either exist within $P_{AS} \cup P_{des}$, or will transition into $P_{AS} \cup P_{des}$. □

4.4.5 Local proof conditions to guarantee that livelocks do not occur

With the conditions from Lemma 3 we ensure that a simplicial active agent will always be present regardless of P_0. We can now introduce Theorem 1, which we use to determine that P_{des} will eventually form from P_0 without livelocks.

Theorem 1 *If the following conditions are satisfied:*

1. P_{des} *is achievable,*
2. *a pattern in* $P \in P_{AS} \cup P_{des}$ *will always be reached from any other pattern* $P \notin P_{AS} \cup P_{des}$,
3. $G^1_{\mathcal{S}}$ *shows that any agent in any state* $s \in \mathcal{S}_{active} \cap \mathcal{S}_{simplicial}$ *can move to explore all open positions surrounding its neighbors (with the exception of when a loop is formed or when it enters a state* $s \in \mathcal{S}_{static}$*),*
4. *in* $G^3_{\mathcal{S}}$*, any agent in any state* $s \in \mathcal{S}_{static}$ *only has outward edges toward states* $s \in \mathcal{S}_{active}$ *(with the exception of a state that is fully surrounded along two or more perpendicular directions),*

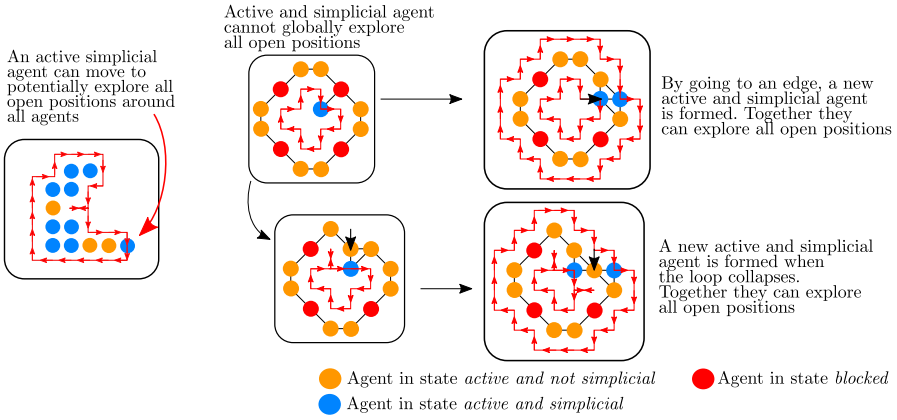
then P_{des} *will always eventually be reached from any initial pattern* P_0.

Proof Consider a swarm of N_{des} agents arranged in a pattern P_0. If P_{des} is achievable, via Lemma 2, P_0 can be composed of any combination of local states without impacting the local ability of the agents to transition into the states $\mathcal{S}_{des}$ (this is the first condition in this theorem). Then, through Lemma 3 we know that if $P_0 \notin P_{AS} \cup P_{des}$, then it will always eventually form a pattern $P \in P_{AS} \cup P_{des}$ (this is the second condition in this theorem). In the following, we will show that any pattern $P \in P_{AS} \cup P_{des}$ will keep transitioning until it forms P_{des}. We observe the case where at least one agent, agent $\mathcal{R}_i$, exists with state $s_i \in \mathcal{S}_{active} \cap \mathcal{S}_{simplicial}$. As agent $\mathcal{R}_i$ moves, one of the following events can happen:

1. Agent $\mathcal{R}_i$ enters a state $s_i' \notin \mathcal{S}_{simplicial}$. Via Lemma 3, at least one other agent is (or will be) in state $s \in \mathcal{S}_{active} \cap \mathcal{S}_{simplicial}$, taking us to point 3 in this list.
2. Agent $\mathcal{R}_i$ enters a state $s_i' \in \mathcal{S}_{static}$. If P_{des} is not yet achieved, then at least one other agent in the swarm is in an active state (Lemma 1). If the active agent(s) are in state $s \in \mathcal{S}_{active} \cap \mathcal{S}_{\neg simplicial}$, then this takes us back to point 1 in this list. If the active agent(s) are in state $s \in \mathcal{S}_{active} \cap \mathcal{S}_{simplicial}$, this takes us to point 3 in this list.
3. Agent $\mathcal{R}_i$, and/or the agent(s) taking over, keeps moving and each time enters a state $s_i' \in \mathcal{S}_{active} \cap \mathcal{S}_{simplicial}$. Via $G^1_{\mathcal{S}}$ we know that it can potentially explore all open positions surrounding all its neighbors (this is the third condition of this theorem). As it moves, its neighbors also change, such that it always can potentially explore all open positions around all agents, and thus *all open positions in the pattern* (see Fig. 8a for a depiction). This means that the swarm can evolve toward a pattern that is closer to the desired one.

Any situation will always develop into the situation of point 3. This is free of livelocks, as all possible livelock situations are mitigated:

1. It may happen that a simplicial and active agent cannot actually visit all open positions in the swarm because, at the global level, it is enclosed in a loop by the other agents. Alternatively, it may happen that it itself creates a loop while moving (this is the first exception to condition 3 of this theorem). By Lemma 3, the loop will always collapse, meaning that a new agent will enter a state $s \in \mathcal{S}_{active} \cap \mathcal{S}_{simplicial}$. The new agent will be able to travel to all positions external to the loop, avoiding a livelock. This resolution is depicted in Fig. 8b.



(a) Simplicial agent that can travel to all open positions in the pattern

(b) Two possibilities for how, should the agent be globally surrounded in a loop and unable to travel to all open positions, then a new simplicial and active agent will take over

Fig. 8 Illustration of how an agent with a state that is active and simplicial can travel to all open positions in the structure

2. Agent $\mathcal{R}_i$ can travel about all open positions in the swarm. Let us assume the extreme case in which $\mathcal{R}_i$ is the only agent that can potentially do this in the entire swarm. Via $G^3_{\mathcal{S}}$, we can verify that, unless P_{des} forms, this must eventually cause at least one static agent to become active (following the fourth condition of this theorem). Consider a static agent $\mathcal{R}_j$ which becomes active when $\mathcal{R}_i$ becomes its neighbor. This may lead to one of the following developments, all of which avoid livelocks.

   (a) Agent $\mathcal{R}_i$ remains in state $s'_i \in \mathcal{S}_{active} \cap \mathcal{S}_{simplicial}$. The pattern can keep evolving further. A livelock is avoided.
   (b) Agent $\mathcal{R}_i$ enters a state $s'_i \in \mathcal{S}_{active} \cap \mathcal{S}_{\neg simplicial}$. By Lemma 3, another simplicial and active agent will be present elsewhere in the swarm. A livelock is avoided.
   (c) As per the second exception to condition 3 of this theorem, agent $\mathcal{R}_i$ enters a state $s \in \mathcal{S}_{static}$ upon becoming a neighbor of agent $\mathcal{R}_j$, before agent $\mathcal{R}_j$ moves. In this case, the departure of agent $\mathcal{R}_j$ will bring it back to a state $s_i \in \mathcal{S}_{active}$ taking us back to points 2(a) or 2(b) in this list.
   (d) Agent $\mathcal{R}_i$ enters a state $s \in \mathcal{S}_{static}$ upon becoming a neighbor of agent $\mathcal{R}_j$, after agent $\mathcal{R}_j$ moves. At this point, either $\mathcal{R}_j$ will move back to its original position and agent $\mathcal{R}_i$ will return to a state $s_i \in \mathcal{S}_{active} \cap \mathcal{S}_{simplicial}$ and keep moving, or $\mathcal{R}_j$ will continue to move elsewhere. In either case, when agent $\mathcal{R}_j$ moves, it will also cause other neighbors to become active. In turn, these will move, and $\mathcal{R}_i$, who also neighbors them, will then return to being in an active state, bringing us back to points 2(a) or 2(b) in this list.
   (e) Agent $\mathcal{R}_i$, after agent $\mathcal{R}_j$ has moved, enters the position (and state) that was originally taken by agent $\mathcal{R}_j$. As in point 2(d) in this list, it is not possible that agent $\mathcal{R}_j$ will always only free $\mathcal{R}_i$ in exactly the same way that agent $\mathcal{R}_i$ freed agent $\mathcal{R}_j$, because $G^3_{\mathcal{S}}$ shows that motions of agent $\mathcal{R}_j$ will free any static agent in the neighborhood, and not just agent $\mathcal{R}_i$.

   There is an exception to the fourth condition of this theorem, which is the static state that is fully surrounded by other agents along two perpendicular axes. In this case, $G^3_{\mathcal{S}}$ may

show that the agent will not directly become active. However, it is trivially impossible (since there is a finite number of agents) for the swarm to only feature agents that are surrounded. A situation where *all* agents are *all* surrounded cannot occur; at least one agent will not be surrounded. This justifies the exception to the fourth condition in this theorem.

With the above it is confirmed that (1) any open position in the pattern can potentially be filled, and (2) no livelocks will arise. This means that the swarm will evolve into all patterns in $P_{AS} \cup P_{des}$. Therefore, P_{des} will always eventually be formed starting from any pattern P_0. □

We thus conclude the proof procedure to check that livelocks will not occur. We showed that by fulfilling a set of local conditions we can determine that the pattern will be achieved from any initial configuration of the swarm. These conditions, being local in nature, are more strict than it is potentially required at the global level. It can be seen that it is actually the agents' ability to stochastically select from a pool of actions that endows them with the potential to keep exploring new neighborhoods and ensure that the swarm keeps evolving without livelocks. A primary condition is the important presence of agents in simplicial active states, which brings interesting insights. Here, we note the following:

- Any desired state with only one neighbor violates the first condition of Lemma 3. This is because this desired state can form the clique of a blocked state on its own. If this occurs, the local conditions are too restrictive to formally guarantee that the swarm will not run into livelocks.
- Removing a dependency on North (Assumption A1) may lead to a violation of the first condition of Lemma 3. This is because desired states become rotation invariant.

4.5 Verifying against the presence of deadlocks

We now have means to verify that no livelocks will occur, but to know that the swarm will always self-organize into the desired pattern, we must also show that no deadlocks can form. That is, there can be no pattern other than the desired pattern P_{des} where none of the agents can take an action. Let us begin, once again, with G_P as introduced in Sect. 4.4. Similarly as to the livelock, we could search exhaustively though G_P for possible nodes with no outgoing edges. Alternatively, we could repeatedly simulate the swarm and experimentally check whether any other pattern forms, but this would not strictly ensure that other patterns cannot manifest.[5] In this work, we still choose to search through G_P. However, to counter the computation explosion, we show that if no livelock exists then it is only necessary to search through a small subset of G_P, and we also provide a method to quickly scan through the remaining subspace (alternatively, if livelocks may exist, then there is technically also no reason to search for deadlocks since we already know that the swarm may evolve undesirably).

4.5.1 Restricting the search space

By definition, deadlocks are patterns $P \neq P_{des}$ where all agents are in a state $s \in \mathcal{S}_{static} = \mathcal{S}_{des} \cup \mathcal{S}_{blocked}$. By Proposition 4 the search space is restricted to patterns that contain at least one agent with state $s \in \mathcal{S}_{des}$.

[5] Considering that the self-organization of the pattern resembles an emergent property, Darley (1994) argues that this would be more efficient.

Proposition 4 *A deadlock cannot consist only of agents with state* $s \in \mathcal{S}_{blocked}$.

Proof Following the same reasoning in Lemma 3, any finite pattern, at its edges, features one of the following:

1. an agent with state $\mathcal{S}_{simplicial}$. By definition, however, $\mathcal{S}_{blocked} \cap \mathcal{S}_{simplicial} = \emptyset$,
2. agents with a state $\mathcal{S}_{\neg simplicial}$ forming a loop boundary. Then, at least one agent must be in a state $\mathcal{S}_{des}$, else it would be in a state $s \in \mathcal{S}_{active}$, which we are not concerned with.

Therefore, in both occurrences, there must be at least one agent with state $s \notin \mathcal{S}_{blocked}$. □

Then, for a certain class of patterns, it can be shown that *all* agents must be in a state $s \in \mathcal{S}_{des}$, as per Proposition 5.

Proposition 5 *If the conditions of Lemma 3 hold and* $\mathcal{S}_{des} \subseteq \mathcal{S}_{simplicial} \cup s_{surrounded}$, *then all agents in a deadlock must be in a state* $s \in S_{des}$.

Proof If $\mathcal{S}_{des} \subseteq \mathcal{S}_{simplicial} \cup s_{surrounded}$, then all states in $\mathcal{S}_{des}$ are either simplicial or $s_{surrounded}$. By the first condition of Lemma 3, none of the states in $\mathcal{S}_{des}$ can satisfy the cliques of any state $\mathcal{S}_{static} \cap \mathcal{S}_{\neg simplicial} - s_{surrounded}$. This means that they cannot ever coexist in the same pattern. By Proposition 4, however, at least one agent must exist with state $s \in \mathcal{S}_{des}$. Therefore, all agents in the spurious pattern must be in a state $s \in \mathcal{S}_{des}$. Alternatively, this proposition can also be verified by a local inspection. □

Therefore, if a pattern is such that $\mathcal{S}_{des} \subseteq \mathcal{S}_{simplicial} \cup s_{surrounded}$, we can further restrict our search to patterns that only have agents in $\mathcal{S}_{des}$. The patterns shown in Fig. 2, with the exception of the hexagon and the line, meet this condition (the line, however, also does not meet Lemma 3).

4.5.2 Finding spurious patterns

In this section we detail our implementation to find spurious patterns for an arbitrary set $\mathcal{S}_{des}$. To sort through the possibilities more efficiently, we analyze state combinations to determine whether they could potentially make a pattern. By first analyzing combinations we need not concern ourselves with the spatial arrangement but only determine whether the states could potentially be combined together independently of order. It is only if such a combination is found that we explore its spatial arrangement, which is done using spanning tree graphs.

Preliminaries Consider a set $\mathcal{S}_{des}$. Because the agents can sense each other omni-directionally, then any two states "match" when two neighbors could have those two states and be neighbors. We introduce two tools to summarize how the states in an arbitrary set $\mathcal{S}_{des}$ match:

– *Match-Direction matrix*, denoted D, is a square matrix $(d \times d)$ that holds the directions along which any two states in $\mathcal{S}_{des}$ are reciprocal to each other.
– *Match-Count matrix*, denoted M, is a square matrix $(d \times d)$ that holds the *number of* directions along which any two states in $\mathcal{S}_{des}$ match. M is symmetric. Intuitively, this is because if agent $\mathcal{R}_i$ is a neighbor of agent $\mathcal{R}_j$, then agent $\mathcal{R}_j$ is a neighbor of agent $\mathcal{R}_i$.

For example, consider the set $\mathcal{S}_{des} = \{s_1, s_2, s_3, s_4\}$ in Fig. 9.

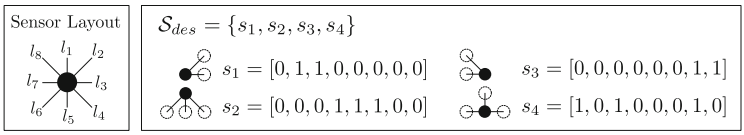


Fig. 9 Set S_{des} used for examples in Sect. 4.5.2. $l_i, i = 1, \ldots, 8$ represent the 8 directions where a neighbor is expected. In a binary representation, if $l_i = 0$ then a neighbor is not expected in that direction, and if $l_i = 1$ then a neighbor is expected. The states s_1,…,s_4 are realizations of this

For this set:

$$D(\mathcal{S}_{des}) = \begin{Bmatrix} - & [l_2] & - & [l_3] \\ [l_6] & - & [l_4] & [l_5] \\ - & [l_8] & - & [l_7] \\ [l_7] & [l_1] & [l_3] & - \end{Bmatrix} \qquad M(\mathcal{S}_{des}) = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

All entries with 0 in $M(\mathcal{S}_{des})$ correspond to empty entries in $D(\mathcal{S}_{des})$. From $M(\mathcal{S}_{des})$ we can quickly extract that state s_1 can never connect to itself or to s_3, but it can connect to states s_2 and s_4.[6] With $D(\mathcal{S}_{des})$ we can see that s_1 can match with s_2 along l_2 and with s_4 along l_3. Note that $D(\mathcal{S}_{des})$, although not strictly symmetric, also has a symmetry to it: each link always features, at its symmetry position, a link along the opposite direction. For example, if s_1 matches with s_2 along direction l_2, then s_2 matches with s_1 along l_6. Therefore, the two matrices essentially provide a local summary of which states can be neighbors and which cannot. This will be used in the following analysis.

Combination analysis A combination of local states should meet a set of conditions independently of how they are arranged. Using these conditions, it is possible to quickly restrict the search space without performing a more computationally expensive spatial analysis. The conditions are:

1. *The topology graph is finite and undirected* For any finite undirected graph $G = (V, E)$, the sum of the vertex degrees must be equal to twice the amount of edges (Van Steen 2010; Ismail et al. 2009). As a consequence, the graph will always feature an even amount of vertices with an odd degree. This is known as the handshaking theorem (Ismail et al. 2009). In our context, this translates to the fact that any valid combination should feature an even amount of states that expect an odd number of neighbors.
2. *The neighbor expectations are reciprocal* In a combination, each state that expects a neighbor in one direction should have at least another state expecting a neighbor in the opposite direction.
3. *The pattern is finite* For each direction, there should be at least one state in a combination that does not expect a neighbor along that direction. Else, the pattern cannot be finite.
4. *The pattern has edges* For each direction, there must be at least one state in the combination that expects a neighbor in that direction, but not in the opposite direction. Otherwise, no state in the combination should expect any neighbor along either direction.
5. *The states can match with each other along all expected directions* Each state in a combination should be capable of being potentially matched (i.e., be a neighbor of) to the

[6] If analyzed visually, s_1 cannot connect to itself because it expects a neighbor to its right (l_3) and top-right (l_2), yet if it were to connect to itself, then the robot next to it would have a neighbor to its left, which thus cannot be s_1. Also, s_1 cannot connect to s_3 because s_1 does not expect a neighbor to be right above itself (at l_1), whereas s_3 would expect a neighbor to be there because it expects a neighbor on its top-left (at l_8), and vice versa.

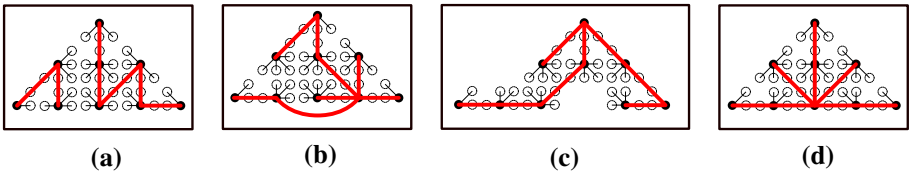


Fig. 10 Examples of: **a** an invalid spanning tree, because the graph is not connected; **b** an impossible spanning tree, because one agent is expected to have more neighbors than it can support; **c** an impossible spanning tree, because some states end up with unfulfilled neighbor expectations; **d** a possible spanning tree

other states in a combination sufficient to cover its expected neighborhood. This information is provided by $M(\mathcal{S}_{des})$ and $D(\mathcal{S}_{des})$. The reasoning is best explained via an example. Consider a swarm of 4 agents with $\mathcal{S}_{des}$ as in Fig. 9 and a potential combination $C_i = \{s_1, s_1, s_2, s_3\}$. Using $M(\mathcal{S}_{des})$, we observe pair-wise matches that are possible between the states in C_i. $M(\mathcal{S}_{des})$ tells us that s_1 only matches with s_2 in one direction. In $D(\mathcal{S}_{des})$ we can see that this is direction l_2 from the perspective of s_1, and l_6 from the perspective of s_2. However, C_i features *two* instances of s_1 and *only one* instance of s_2. This means that one instance of s_1 can never be satisfied—the combination cannot exist. This can be checked for all states.

Spanning trees analysis Combinations that have the potential to form a pattern are analyzed further. We do this by composing spanning tree graphs. Let $T_i(C_k)$ represent an arbitrary spanning tree generated from a combination C_k. The nodes of T_i are the states in C_k, and the edges of T_i are one of the connections between the states. A representative spanning tree must meet the conditions below.

- It is acyclic.
- It is simple (no duplicate edges).
- The edges must at least meet the match conditions in $M(\mathcal{S}_{des})$, or else we know that the edges are impossible because the two states can never be neighboring states.
- It is connected. If operating by Π_f, then the swarm is connected. This means that it can be represented by a connected spanning tree. If $T_i(C_k)$ is not connected, as in the example in Fig. 10a, then it is invalid.
- The degree of each state should be less than or equal to the number of neighbors that an agent in that state expects. If the degree of a node in $T_i(C_k)$ is larger than the degree of the state, then $T_i(C_k)$ is invalid and the spanning tree is discarded, as for the example in Fig. 10b.
- The spatial arrangement must be feasible. All other conditions above depend on the properties of the spanning tree graph and are not (directly) dependent on the spatial arrangement of the states. In this last condition, we analyze the spatial arrangement of the graph to see if all neighboring states match without lose ends (i.e., "unfulfilled neighbors"), or loops where two states are eventually expected to occupy the same positions. For instance, Fig. 10c shows a spanning tree that fails this test. $D(\mathcal{S}_{des})$ can be used to quickly generate the full pattern.

If a possible spanning tree is found, as in Fig. 10d, then a possible pattern has been identified and it can be checked to determine whether it is equivalent to P_{des} or whether it is spurious. A variety of methods can be used to do so automatically (Loncaric 1998).

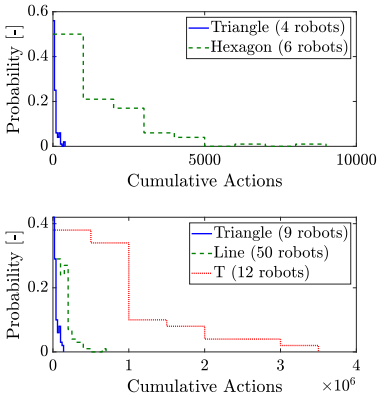


Fig. 11 Normalized histograms of the actions taken before the pattern is achieved for the different patterns tested. The plots are separated in two for scale differences. The bin width was adjusted to show the overall trend for each pattern. The cross with 20 robots is excluded for scale reasons, with the lowest amount of steps measured being $\approx 2.5 * 10^5$

5 Evaluation of the idealized system

We begin by evaluating the performance of the idealized swarm as described in Sect. 4.1. This allows us to investigate more fundamental properties and gain initial high-level insights. We also explore how further tuning of Π_f could affect the statistical performance of the swarm in forming a desired pattern more quickly. The latter leads to insights on possible optimization strategies, which we discuss further in Sect. 7.3.

The simulation environment used in this section replicates the idealized framework from Sect. 4.1. We simulated idealized agents on a discrete 2D grid world operating in discrete time. At each time step, one random agent with state $s \in \mathcal{S}_{active}$ executes an action based on Π_f. All tests begin by initializing the agents in a random pattern P_0 and end when all agents are in a state $s \in \mathcal{S}_{static}$.

We evaluated the formation of the patterns from Fig. 2. All patterns were successfully achieved, with no collisions or separation ever occurring. This also happened for the line, which did not pass the proof and was additionally also prone to spurious patterns. Generally, as the complexity of the pattern and size of the swarm grew, the cumulative actions taken by the swarm to go from P_0 to P_{des} also grew significantly. The swarm is successfully safe and forms the desired patterns, even though (as expected due to the low cognition of the robots) it can take a significant amount of steps before the swarm self-organizes into the pattern. This can be appreciated in the histograms of the results shown in Fig. 11, split in two graphs to address the difference in scale. Note that the line with 50 robots performed better than the T with only 12 robots. When we also analyze the mean number of actions per agent, we see that the histograms of the line with 50 robots and the triangle with 9 robots are comparable. This implies that there is a deeper correlation with shape complexity that should be explored further.

Motivated by the increasingly low performance of larger and/or more complex patterns, we explored certain alterations of the behavior in order to investigate whether it was possible to achieve the pattern faster than in the baseline tests above. We tested this for the triangles with 4 and 9 robots and the hexagon, for which the expected number of actions were fewer and the differences could be better investigated. We explored the following alterations:

- *Alteration 1 (ALT1)*: same as baseline; however, when an agent moves at time step k, the same agent will not move at time step $k + 1$ (unless it is the only active agent).
- *Alteration 2 (ALT2)*: same as ALT1; additionally, all states with more than 5 neighbors are now not mapped to any actions.

– *Alteration 3 (ALT3)*: same as ALT2; additionally, all actions must ensure that all agents in the neighborhood, following the action, have at least one neighbor at North, South, East or West, else the state-action pair is discarded from Π_f. For the triangle with 9 agents, we made one exception to this, and it is the state $s = [1\ 0\ 1\ 0\ 0\ 0\ 1\ 0]$ (following the layout in Fig. 9) for which otherwise a spurious pattern could also form.
– *Alteration 4 (ALT4)*: same as ALT3; additionally, all states with more than 4 neighbors are now not mapped to any actions.

ALT3 and ALT4 stem from the intuition to let agents "cut-corners" and have fewer functionally active states. In turn, however, ALT3 and ALT4 do no meet the conditions of Lemma 2 for the hexagon of 6 robots, and do not meet Condition 3 of Theorem 1 for all patterns tested with it. This is because some states in $\mathcal{S}_{simplicial}$ lost their property of enabling the agent to potentially move freely around its neighborhood. Functionally, they behaved like states in the set $\mathcal{S}_{\neg simplicial}$, and a few even like states in $\mathcal{S}_{blocked}$. Normalized distributions for the number of steps to completion using ALT1-ALT4 are shown in Fig. 12a–c for the triangle with 4 agents, the hexagon, and the triangle with 9 agents, respectively. For ALT1 and ALT2 the final pattern is achieved in all cases. As the size of the pattern grows, ALT1 and ALT2 are seen to provide a marginally better performance, but not significantly so. The real improvement is seen with ALT3 and ALT4. By blocking more local states and cutting corners, the swarm is less chaotic and forms the pattern orders of magnitude faster. As expected through Lemma 2, however, ALT3 and ALT4 prevented the hexagon from forming. Instead, failing condition 3 of Theorem 1 did not stop ALT3 and ALT4 from achieving the triangles with 4 and 9 robots. This could imply that Theorem 1, by nature of featuring local conditions, becomes more restrictive than necessary for some global patterns. This was also the case for the line with 50 agents, because the line also does not meet the condition. Alternatively, it could also be possible that the robots were simply "lucky" to not encounter deadlock situations during any of our simulations.

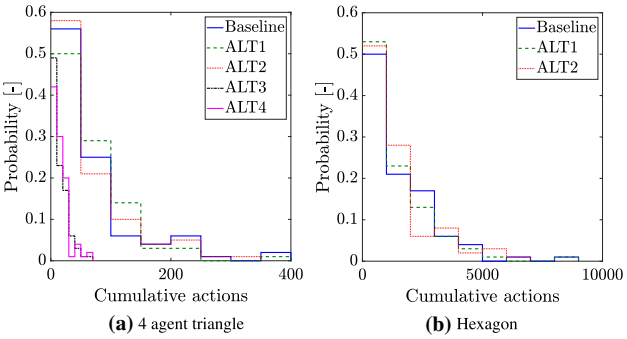
6 Implementing the behavior on robots

Until now, we have dealt with idealized agents on a 2D grid. In this section, we describe how the behavior can be brought to real robots operating in continuous time and space and using local clocks. We test the behavior in two stages of fidelity: (1) accelerated particles, and (2) simulated MAV flights, showing that the behavior is also robust to noise.

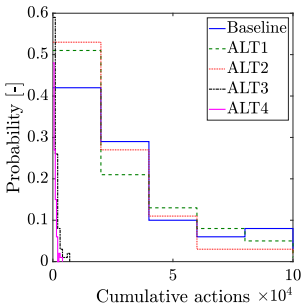
6.1 Robot behavior

The robots can sense omni-directionally all their neighbors within a range ρ_{sensor} and can determine whether their neighbors are computing an action (Assumption A3). A robot $\mathcal{R}_i$ determines its discrete local state $s_i \in \mathcal{S}$ following the bearing based discretization in Fig. 13a.

All robots act following the FSM in Fig. 13b. This FSM locally enforces that only one robot in the neighborhood can move at any time. Following this FSM, a robot will initiate and pursue an action from Π_f if and only if no other robot in a neighborhood is sensed to be already doing so, which locally recreates the conditions of the idealized system. Therefore, even though multiple robots around the swarm can take actions at the same time, this does not occur at the local level. If two robots who are not neighbors become neighbors while both are executing an action, the actions will interrupt, ensuring safety. Using $t_{adj} > 0$ and $t_{wait} > 0$ the robots have allocated time to execute attraction, repulsion, and alignment behaviors. As



(a) 4 agent triangle **(b)** Hexagon



(c) 9 agent triangle

Fig. 12 Normalized histograms of actions to completion by different alternations of the state-action spaces for three patterns. The bin width was adjusted to show the overall trend for each case

these alignments maneuvers are minimal, they are not sensed by neighbors as actions and therefore create natural time windows whereby robots take turns in taking actions.

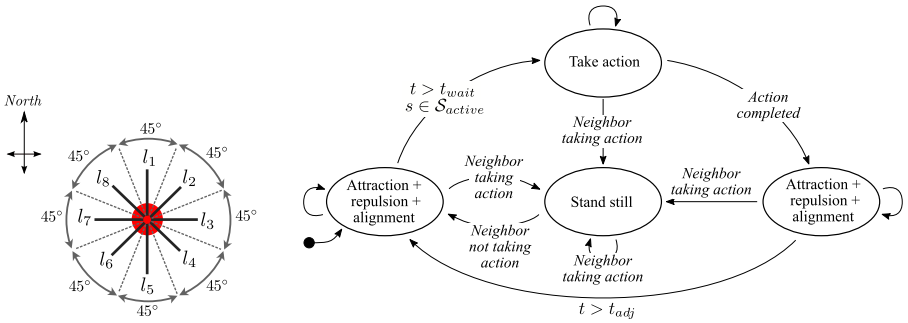
We have designed a unified attraction, repulsion, and alignment behavior that allows the robots to naturally arrange in a grid structure whenever not executing an action. Consider a robot $\mathcal{R}_i$ and its neighbor $\mathcal{R}_j$. The robots are controlled according to a North-East (NE) frame of reference. The commanded velocity of $\mathcal{R}_i$ along North (and, equivalently, East) is given by:

$$v_{N_{cmd_{ij}}} = \underbrace{(v_{r_{ij}} + v_{b_{ij}})\cos(\beta_{ij})}_{\text{Attraction and repulsion}} - \underbrace{v_{b_{ij}}\cos(2\beta_{des} - \beta_{ij})}_{\text{Alignment}}. \tag{3}$$

The first term handles attraction and repulsion. The second term aligns $\mathcal{R}_i$ at a bearing β_{des} to $\mathcal{R}_j$. β_{ij} is the bearing of $\mathcal{R}_i$ to $\mathcal{R}_j$ with respect to North. $v_{b_{ij}}$ is the desired radial velocity. The attraction–repulsion velocity $v_{r_{ij}}$ is:

$$v_{r_{ij}} = -k_r \frac{1}{|\rho_{ij}|} + \frac{1}{1 + e^{-k_a(|\rho_{ij}| - \rho_s)}}, \tag{4}$$

where $k_r \geq 0$ is the repulsion gain, $k_a \geq 0$ is the attraction gain, ρ_{ij} is the range between $\mathcal{R}_i$ and $\mathcal{R}_j$, ρ_s is a shift in the attraction term used to tune the equilibrium distance to ρ_{des}. Equation 4 has Lyapunov stability (Gazi and Passino 2002). For given ρ_{des}, k_r, and k_a, one can extract ρ_s such that $v_{r_{ij}} = 0$. The two robots are in equilibrium ($v_{N_{cmd_{ij}}} = v_{E_{cmd_{ij}}} = v_{N_{cmd_{ji}}} = v_{E_{cmd_{ji}}} = 0$) when $\beta_{ij} = \beta_{des}$, $\beta_{ji} = \beta_{des} \pm \pi$, and $v_{r_{ij}} = v_{r_{ji}} = 0$. Note that Eq. 3 is reciprocal. For each β_{des}, there exists a corresponding equilibrium point at $\beta_{des} \pm \pi$.



(a) Rounding method used by the robots to discretize their local state

(b) FSM of robot behavior

Fig. 13 State discretization and FSM of robot behavior

This is due to the identities $\sin(\beta+\pi) = -\sin(\beta)$ and $\cos(\beta+\pi) = -\cos(\beta)$. Furthermore, multiple desired bearings β_{des} can be defined, such that each robot can gravitate to the one that is closest to its current β. We provided the robots with $\beta_{des} = \{0, \pi/4, \pi/2, 3\pi/4\}$, making them adjust all the 8 bearings to each other that match the idealized grid. For $\beta_{des} = \pi/4$ and $\beta_{des} = 3\pi/4$, then we define $\rho_{des} = \sqrt{2}$m instead of $\rho_{des} = 1$m. For a robot $\mathcal{R}_i$ which senses m neighbors, the complete command along North is $v_{N_{cmd_i}} = \sum_{j=1}^{m} v_{N_{cmd_{ij}}}$, and the equivalent for East. This is unless the closest neighbor is at a distance $\rho < \rho_{safe}$, in which case only the closest neighbor is considered.

6.2 Simulation tests with accelerated particles

We begin by testing the behavior from Sect. 6.1 on accelerated particles in an unbounded 2D space. This allows us to quickly test the performance of large swarms while remaining independent of the dynamics of any particular robot.

The simulations in these sections have been executed on an in-house simulator called *Swarmulator*. Swarmulator is a light-weight swarm simulator designed to quickly develop and prototype spatial swarm behavior.[7] Swarmulator's simplicity and emphasis on quick prototyping is the reason that it was chosen for this phase. Each robot is simulated as a point in an unbounded 2D space by a detached C++ thread, thus simulating a random asynchronicity and minimizing the simulation artifact that would otherwise stem from simulating the swarm in a loop. To further reduce simulation artifacts, the robots initiate the behavior with a random local time $0 < t < t_{wait}$. Other detached threads handle animation and logging, allowing automatic checks of global properties. In the simulations: $\rho_{sensor} = 1.6$m, $\rho_{des} = 1$m, $\rho_{safe} = 0.5$m, $t_{adj} = 1.8$s, $t_{wait} = 3.6$s, $k_r = 1$, $k_a = 5$, $v_{action} = 1$m/s, $v_b = 10$m/s. The state-action map Π_f is as in ALT4 from Sect. 5.

Results The results for the triangles with 4 and 9 agents from Sect. 5, using the controller from ALT4, were validated in this continuous setting. Figure 14a, b shows sample trajectories over time.[8] We can see that the agents reshuffle until the desired pattern is achieved. All

[7] The source code can be found at https://github.com/coppolam/swarmulator/tree/SI_PatternFormation.

[8] Videos of other sample runs are available at https://www.youtube.com/playlist?list=PL_KSX9GOn2P8BYpwA-_WfXmtb7CRnVhC3.

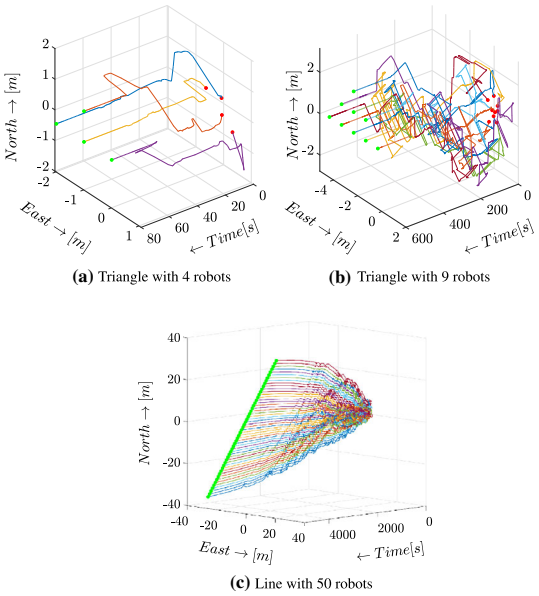


Fig. 14 Simulated trajectories to the desired patterns

(a) Triangle with 4 robots

(b) Triangle with 9 robots

(c) Line with 50 robots

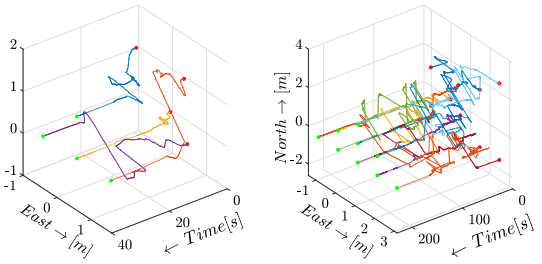
simulations were repeated 50 times. The triangle with 4 agents was achieved successfully in 50 out of 50 trials, with generally fast convergence times (within 100 seconds of simulated time). The triangle with 9 agents was achieved successfully in 49 out of 50 trials. Only one trial experienced a separation. This happened as two non-neighboring agents chose to perform an action at approximately the same time, came into each other's view, but the alignment maneuvers that followed were such that two agents (who were the link between two parts of the swarm) momentarily moved further than 1.6m apart, which was the limit of the sensor. Although we could be more lenient and accept the fact that the swarm quickly reconnects, as done by Winfield and Nembrini (2012), the issue is noted and should be tackled in future work to further guarantee safety even in a continuous setting. Nevertheless, this was *the only* "unsafe" event that took place out of thousands of maneuvers executed over all 50 trials. We also successfully simulated the behavior of the swarm with large groups tasked with making a line with 50 robots, for which a sample trajectory is shown in Fig. 14c. Here, it is interesting to see how the line slowly forms as robots all over the swarm begin to align themselves as required.

6.3 Micro air vehicle simulations

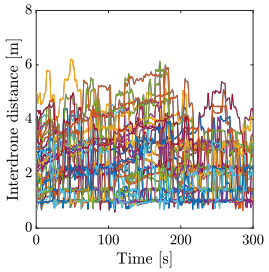
Having developed and tested a behavior that can be used in a continuous domain, we now explore whether it can be used when robots have more realistic dynamics and reaction times. This section provides a proof of concept and shows how the selected algorithm can work on a team of real MAVs with the relevant dynamic constraints and perturbations.

The simulations were executed using Robot Operating System (ROS) (Quigley et al. 2009) and the Gazebo physics engine (Koenig and Howard 2004). The hector-quadrotor model provided by Meyer et al. (2012) simulates the dynamics of a quad-rotor MAV. Each MAV is simulated on a separate module and runs independently, with the higher-level controller running at 10 Hz. The same simulation environment was used in both Coppola et al. (2018)

Fig. 15 Exemplary results of ROS simulations



(a) Triangle with 4 robots **(b)** Triangle with 9 robots



(c) Simulated flight of 9 MAVs with sensor noise

and McGuire et al. (2017) with successful replication of the controllers on real-world MAVs, and it was chosen for this reason. We assumed that the MAVs could measure the position of their nearest neighbors up to 1.6 m, and that they could then sense whenever a neighbor was moving at more than 0.1 m/s, which they would interpret as the neighbor taking an action. All other control parameters were kept the same as in the Swarmulator trials, with some minimal tuning to suit the new dynamics (namely: $v_b = 2$, $t_{adj} = 1.5$, $t_{wait} = 3$).

Results The results of Sect. 6 were successfully replicated using this set-up. We show two sample trajectories of flights in Fig. 15a, b.[9] As for the accelerated particles, the triangle with 4 MAVs was generally reached within only 100 s of flight, and in 48 out of 50 cases it was completed before the final simulation time of 500 s. As expected based on our idealized simulation, the flight time was not enough for such a high success rate also with the significantly more complex triangle with 9 MAVs. 20 out of 50 cases finished the triangle within the maximum simulation time of 5000 s for these simulations. Nevertheless, the MAVs never collided with each other and the swarm never separated in any of the trials, showing that the idealized rules translate well to realistic dynamics.

Simulation results with sensor noise Additionally, we explored the performance of the behavior under the influence of noise in the relative position readings of neighbors by applying Gaussian noise with standard deviation of 0.1 m and 0.1 rad for relative range and bearing, respectively. The only change was that the MAVs could see up to 2 m instead of 1.6 m in order to restrict false negatives. The results were robust to the noise. Consider, for instance, the 300 s flight with 9 MAVs shown in Fig. 15c. It can be seen that the swarm distances are kept, while the swarm still reshuffles, and no collisions occur. The discretization imposed by the state-action map is such that the behavior is robust to sensor noise. The behavior is

[9] Videos are available at https://www.youtube.com/playlist?list=PL_KSX9GOn2P8BYpwA-_WfXmtb7CRnVhC3.

(a) Desired pattern and related desired states

(b) Asymmetric triangle in ROS simulation

(c) Flight path

Fig. 16 Development of an asymmetric triangle and test on ROS

robust even when the same set-up from the noiseless case is used, without any filtering of the Gaussian noise (e.g., using a Kalman filter or a low pass filter), which would otherwise drastically improve the results further.

7 Discussion

7.1 Intuitive and verifiable design of complex behaviors

The approach presented in this paper allows a swarm designer to intuitively define local behavior of cognitively limited robots faced with a global task. It is merely necessary to divide the global task into its locally observable constituents and incorporate this into the state-action map of the robots. Doing so provides the robots with a behavior that forms the pattern, even though the robots are incapable of locally knowing when/if this ever occurs.

Having such an intuitive method allows us to form patterns that (for systems with similarly limited capabilities) had previously not been achieved using an explicit design. In this paper we showed six patterns as examples, but the limits of the algorithm do not stop there. Izzo et al. (2014) and Scheper and de Croon (2016), for instance, both proposed neural networks to tackle the formation of an asymmetric triangle, whereby the difficulty was that three non-communicating homogeneous robots could not resolve the asymmetry. However, using the approach presented in this paper, it becomes easy to form any asymmetric triangle. The desired states to develop Π_f are readily extracted, as in Fig. 16a, and the dimensions of the triangle can be tuned by adjusting the attraction and repulsion forces along North and East. The asymmetric triangle is then obtained as exemplified in Fig. 16b, c.

In this work we focused on pattern formation, but we postulate that this framework could also be extended to other global tasks such as organized navigation or task allocation. In future work, we aim to investigate how the framework can be generalized.

7.2 Generating arbitrary patterns without livelocks and deadlocks

Section 4.2 showed how Π_f can be readily computed for any pattern. However, because of how limited the robots are, it is not necessary that the swarm is able to reach this pattern from any initial condition while being free of livelocks or deadlocks. Deadlocks and livelocks, however, stem from the limited knowledge that is available to the robots. If the robots could see

further, or remember past states, or communicate, they would be able to form more complex patterns and would be able to move more freely. Theoretically then, any pattern can be formed provided that the state space is sufficiently detailed to uniquely represent the desired goal and allow enough freedom to the robots. In line with the goals of generalizing the scheme that was presented here, we also wish to determine how providing the agents with some extra capabilities can allow more complex goals to emerge. This is while resting on the knowledge that the swarm can also operate when these extra capabilities malfunction. Furthermore, the proof conditions presented in this paper have been shown to be more restrictive than it can turn out to be in a real swarm. The advantage of using local properties are that we do not need to analyze the global states of the swarm, yet this comes at the cost of possibly being more strict than required from the global perspective. At this moment, however, we have seen that patterns that do not respect some of the conditions still form in our simulations, such as the line pattern. Indeed, it may be that the subset of global states that represent a deadlock or livelock is very small compared to the total state space, making such failures possible, yet extremely unlikely. More focused investigations should be conducted in order to understand when it is possible to be more lenient on some conditions while still ensuring that livelocks and deadlocks do not arise.

7.3 Time for self-organization

In Sect. 5, generally speaking, it was found that as the size of the swarm and the intricacy of the pattern grow, the pattern could form only after a possibly unrealistic number of actions by the robots. This property was expected in light of all the limitations of the robots, as it becomes increasingly unlikely that the agents' random actions will lead to the desired global pattern.[10] However, there are two important things to note:

1. In real robot swarms, several actions will be taking place at the same time, so the time to completion will be faster than expected. For instance, in Fig. 14c the line is seen to slowly form across the entire swarm, whereas this is not the case for the idealized system.
2. Our investigations in Sect. 5 showed that it is possible to improve performance by several orders of magnitude by further altering the local state-action map.

The latter leads to questions about how to best alter a local state-action map. The alterations in this work were done manually, using intuition, for exploratory purposes. The problem could be solved more optimally using machine learning methodologies such as reinforcement learning or evolutionary robotics. The objective would be to alter Π_f such that, statistically, the time for the robots to self-organize into a desired pattern is minimized. Here, the local proofs would allow us to verify that the alterations are such that the system is still guaranteed, at all times, to always eventually reach the pattern.

7.4 Toward real-world implementations and applications

The simulations using ROS in Sect. 6.3 provide a large degree of confidence in the possibility to implement the system on real MAVs (or other robots). Provided that the necessary sensory information is available, then they are able to follow the behavior even when behaving by their own internal clock and in the presence of sensor noise, and this is without the aid of any additional filtering. We then find that the local behavior can also be used simply to guarantee

[10] In popular adage, one might say that there is no such thing as a free lunch.

collision avoidance and swarm coherence in spite of all limitations of the robots. This has several applications of its own. For instance, it can be used to preemptively guarantee that a robotic sensor network never separates in multiple groups.

7.5 Scalability of proof procedure

Our proof procedure focused as much as possible on the local level, making it largely independent of the number of agents in the swarm, and thus able to mitigate state explosion issues. Most notably, we are able to prove, only by a local-level analysis, that livelocks will not exist when starting from any initial pattern. A key element of this proof was an analysis of the simplicial states and the intuition that they could help the swarm to resolve livelocks. Nevertheless, the complete proof still requires us to verify that deadlock patterns will not occur, and this part is still done using an ultimately global analysis. We have shown how to mitigate the computational explosion by looking at a limited subset of state combinations and using a procedure to quickly sort through the possibilities, yet the issue is not yet fully eliminated. In future research, there should be efforts to further mitigate its effects for finite patterns. Here, we expect that the match matrices introduced will be a fundamental tool to analyze local connections between the robots.

For now, three solution directions have been identified in order to mitigate the computational explosion. The first is to focus on the agents at the border of the structure, assuming that all other agents will be enclosed by these agents. The second avenue is to use repeating sub-patterns. The local states could be made such that the agents can arrange into infinitely repeating patterns (e.g., infinitely connecting hexagons) and create a large complex structure without defining or checking the larger structure in full. This we actually already did, in part, for the line pattern. The third solution, perhaps most trivial, is to allow robots that have been blocked for a long time to temporarily perform partially unsafe maneuvers, which might set the system free from deadlocks (but may come at other costs).

8 Conclusion and future work

In this paper we introduced a method to design the local behavior of robots in a swarm so as to form desired global patterns in spite of extremely limited cognitive abilities. Because the robots only know the relative location of their closest neighbors and have no memory of the past, they cannot take “purposeful” actions. Therefore, a mechanism has been designed that makes the global pattern emerge from the local, stochastic behaviors of the agents. Approaching the problem from top-down, the method simply requires one to identify the local states that build the desired global pattern in order to design the behavior. Then, to close the loop, we presented a proof procedure to verify whether the desired pattern will always eventually emerge from the stochastic interactions of the agents. An important insight from these proofs is the crucial role that simplicial states play in helping the swarm to avoid livelocks and minimizing the possibility of deadlocks. It is important to note here, however, that should we find that livelocks and deadlocks are possible, then this tells us that the robots have an insufficient sensory knowledge for the desired global goal to always eventually happen, which is equally valuable information when designing a robotic swarm. Despite developing the behavior for idealized agents on a grid world, we have shown very promising results that show that the behavior can be successfully reproduced by robots operating in continuous time and space, with local clocks, even in the presence of noise.

The methodology presented here has been used for pattern formation. At its core, however, it is based on the more general idea of synthesizing a global goal into a probabilistic state-action map executed by the robots, and the verification of the global property by ensuring that the swarm features agents with a state that empowers them to help the swarm evolve (i.e., simplicial states). With a modified mapping, we expect this strategy to also be applicable to systems with significantly different state and action spaces. Furthermore, following the positive results of the simulations presented in this paper, future work will focus on bringing this framework to real-world robots. A primary challenge that must be solved for this to happen is to use an optimization procedure to enable larger and more complex patterns to form faster. A second challenge is to explore the best ways of dealing with potential false positives or false negatives. These situations may cause the robot to take a misguided action. We expect that this can be solved by further limiting the state-action map whenever a state cannot be clearly identified. Finally, it would be valuable to investigate the impact of removing the knowledge of a common North direction on the ability to create certain patterns, making the system even more independent from the environment.

References

Achtelik, M., Achtelik, M., Brunet, Y., Chli, M., Chatzichristofis, S., Decotignie, J. D., et al. (2012). SFly: Swarm of micro flying robots. In *2012 IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 2649–2650). Washington: IEEE Press.

Arbuckle, D. J., & Requicha, A. A. G. (2010). Self-assembly and self-repair of arbitrary shapes by a swarm of reactive robots: Algorithms and simulations. *Autonomous Robots*, *28*(2), 197–211.

Arbuckle, D. J., & Requicha, A. A. G. (2012). Issues in self-repairing robotic self-assembly. In R. Doursat, H. Sayama, & O. Michel (Eds.), *Morphogenetic engineering: Toward programmable complex systems* (pp. 141–155). Berlin: Springer.

Basiri, M., Schill, F., Floreano, D., & Lima, P. U. (2014). Audio-based localization for swarms of micro air vehicles. In *2014 IEEE international conference on robotics and automation (ICRA)* (pp. 4729–4734). Washington: IEEE Press.

Bonabeau, E., & Dessalles, J.-L. (1997). Detection and emergence. *Intellectica*, *2*(25), 85–94.

Bonabeau, E., Guérin, S., Snyers, D., Kuntz, P., & Theraulaz, G. (2000). Three-dimensional architectures grown by simple 'stigmergic' agents. *Biosystems*, *56*(1), 13–32.

Clarke, E. M, Jr., Grumberg, O., & Peled, D. A. (1999). *Model checking*. Cambridge, MA: MIT Press.

Conroy, J., Samuel, P., & Pines, D. (2005). Development of an MAV control and navigation system. In *Infotech@ Aerospace, AIAA 2005, Arlington, Virginia* (p. 7065).

Coppola, M., McGuire, K. N., Scheper, K. Y. W., & de Croon, G. C. H. E. (2018). On-board communication-based relative localization for collision avoidance in micro air vehicle teams. *Autonomous Robots*, *42*(8), 1787–1805.

Darley, V. (1994). Emergent phenomena and complexity. *Artificial Life*, *4*, 411–416.

de Marina Peinado, H. J. G. (2016). *Distributed formation control for autonomous robots*. Groningen: University of Groningen.

Derakhshandeh, Z., Gmyr, R., Richa, A. W., Scheideler, C., & Strothmann, T. (2016). Universal shape formation for programmable matter. In *Proceedings of the 28th ACM symposium on parallelism in algorithms and architectures (SPAA '16)* (pp. 289–299). New York, NY: ACM.

Di Luna, G. A., Flocchini, P., Santoro, N., Viglietta, G., & Yamauchi, Y. (2017). Shape formation by programmable particles. ArXiv Preprint. arXiv:1705.03538.

Dixon, C., Winfield, A. F. T., Fisher, M., & Zeng, C. (2012). Towards temporal verification of swarm robotic systems. *Robotics and Autonomous Systems*, *60*(11), 1429–1441.

Engelen, S., Gill, E. K. A., & Verhoeven, C. J. M. (2011). Systems engineering challenges for satellite swarms. In *2011 aerospace conference, AERO '11* (pp 1–8). Washington, DC: IEEE Computer Society.

Faigl, J., Krajník, T., Chudoba, J., Přeučil, L., & Saska, M. (2013). Low-cost embedded system for relative localization in robotic swarms. In *2013 IEEE international conference on robotics and automation (ICRA)* (pp. 993–998). Washington: IEEE Press.

Falconi, R., Gowal, S., & Martinoli, A. (2010). Graph based distributed control of non-holonomic vehicles endowed with local positioning information engaged in escorting missions. In *2010 IEEE international conference on robotics and automation (ICRA)* (pp. 3207–3214). Washington: IEEE Press.

Falconi, R., Sabattini, L., Secchi, C., Fantuzzi, C., & Melchiorri, C. (2011). A graph-based collision-free distributed formation control strategy. In *IFAC proceedings volumes, 18th IFAC world congress* (Vol. 44(1), pp. 6011–6016).

Falconi, R., Sabattini, L., Secchi, C., Fantuzzi, C., & Melchiorri, C. (2015). Edge-weighted consensus-based formation control strategy with collision avoidance. *Robotica*, *33*(2), 332–347.

Flocchini, P., Prencipe, G., Santoro, N., & Widmayer, P. (2005). Gathering of asynchronous robots with limited visibility. *Theoretical Computer Science*, *337*(1), 147–168.

Fox, M. J., & Shamma, J. S. (2015). Probabilistic performance guarantees for distributed self-assembly. *IEEE Transactions on Automatic Control*, *60*(12), 3180–3194.

Gazi, V., & Passino, K. M. (2002). A class of attraction/repulsion functions for stable swarm aggregations. In *Proceedings of the 41st IEEE conference on decision and control (CDC)* (Vol. 3, pp. 2842–2847).

Gazi, V., & Passino, K . M. (2004). Stability analysis of social foraging swarms. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *34*(1), 539–557.

Gerkey, B. P., & Matarić, M. J. (2004). A formal analysis and taxonomy of task allocation in multi-robot systems. *The International Journal of Robotics Research*, *23*(9), 939–954.

Gjondrekaj, E., Loreti, M., Pugliese, R., Tiezzi, F., Pinciroli, C., Brambilla, M., et al. (2012). Towards a formal verification methodology for collective robotic systems. In T. Aoki & K. Taguchi (Eds.), *Formal methods and software engineering: 14th international conference on formal engineering methods (ICFEM), Kyoto, Japan, November 12–16, 2012. Proceedings* (pp. 54–70). Berlin: Springer.

Grushin, A., & Reggia, J. A. (2008). Automated design of distributed control rules for the self-assembly of prespecified artificial structures. *Robotics and Autonomous Systems*, *56*(4), 334–359.

Grushin, A., & Reggia, J . A. (2010). Parsimonious rule generation for a nature-inspired approach to self-assembly. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, *5*(3), 12:1–12:24.

Guo, K., Qiu, Z., Meng, W., Xie, L., & Teo, R. (2017). Ultra-wideband based cooperative relative localization algorithm and experiments for multiple unmanned aerial vehicles in GPS denied environments. *International Journal of Micro Air Vehicles*, *9*(3), 169–186.

Haghighat, B., & Martinoli, A. (2017). Automatic synthesis of rulesets for programmable stochastic self-assembly of rotationally symmetric robotic modules. *Swarm Intelligence*, *11*(3), 243–270.

Hamann, H. (2018). *Swarm robotics: A formal approach*. Berlin: Springer.

Ismail, A. S., Hasni, R., & Subramanian, K. (2009). Some applications of eulerian graphs. *International Journal of Mathematical Science Education*, *2*(2), 1–10.

Izzo, D., & Pettazzi, L. (2005). Equilibrium shaping: Distributed motion planning for satellite swarm. In *Proceedings of the 8th international symposium on artificial intelligence, robotics and automation in space*.

Izzo, D., & Pettazzi, L. (2007). Autonomous and distributed motion planning for satellite swarm. *Journal of Guidance, Control, and Dynamics*, *30*(2), 449–459.

Izzo, D., Simões, L. F., & de Croon, G. C. H. E. (2014). An evolutionary robotics approach for the distributed control of satellite formations. *Evolutionary Intelligence*, *7*(2), 107–118.

Ji, M., & Egerstedt, M. (2007). Distributed coordination control of multiagent systems while preserving connectedness. *IEEE Transactions on Robotics*, *23*(4), 693–703.

Joordens, M. A., & Jamshidi, M. (2010). Consensus control for a system of underwater swarm robots. *IEEE Systems Journal*, *4*(1), 65–73.

Klavins, E. (2002). Automatic synthesis of controllers for distributed assembly and formation forming. In *2002 IEEE international conference on robotics and automation (ICRA)* (Vol. 3, pp. 3296–3302). Washington: IEEE Press.

Klavins, E. (2007). Programmable self-assembly. *IEEE Control Systems*, *27*(4), 43–56.

Koenig, N., & Howard, A. (2004). Design and use paradigms for Gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ international conference on intelligent robots and systems (IROS)* (vol. 3, pp. 2149–2154). Washington: IEEE Press.

Konur, S., Dixon, C., & Fisher, M. (2012). Analysing robot swarm behaviour via probabilistic model checking. *Robotics and Autonomous Systems*, *60*(2), 199–213.

Krishnanand, K. N., & Ghose, D. (2005). Formations of minimalist mobile robots using local-templates and spatially distributed interactions. *Robotics and Autonomous Systems*, *53*(3), 194–213.

Lerman, K., Galstyan, A., Martinoli, A., & Ijspeert, A. (2001). A macroscopic analytical model of collaboration in distributed robotic systems. *Artificial Life*, *7*(4), 375–393.

Loncaric, S. (1998). A survey of shape analysis techniques. *Pattern Recognition*, *31*(8), 983–1001.

McGuire, K. N., Coppola, M., de Wagter, C., & de Croon, G. C. H. E. (2017). Towards autonomous navigation of multiple pocket-drones in real-world environments. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 244–249). Washington: IEEE Press.

McGuire, K. N., de Croon, G. C. H. E., de Wagter, C., Remes, B., Tuyls, K., & Kappen, H. (2016). Local histogram matching for efficient optical flow computation applied to velocity estimation on pocket drones. In *2016 IEEE international conference on robotics and automation (ICRA)* (pp. 3255–3260). Washington: IEEE Press.

Mesbahi, M., & Egerstedt, M. (2010). *Graph theoretic methods in multiagent networks* (Vol. 33). Princeton: Princeton University Press.

Meyer, J., Sendobry, A., Kohlbrecher, S., Klingauf, U., & von Stryk, O. (2012). Comprehensive simulation of quadrotor UAVs using ROS and Gazebo. In I. Noda, N. Ando, D. Brugali, & J. J. Kuffner (Eds.), *J. Simulation, modeling, and programming for autonomous robots* (pp. 400–411). Berlin: Springer.

Nembrini, J., Winfield, A., & Melhuish, C. (2002). Minimalist coherent swarming of wireless networked autonomous mobile robots. In B. Hallam, D. Floreano, J. Hallam, G. Hayes, & J.-A. Meyer (Eds.), *From animals to animats 7: Proceedings of the seventh international conference on simulation of adaptive behavior, ICSAB* (pp. 373–382). Cambridge, MA: MIT Press.

Oh, K.-K., Park, M.-C., & Ahn, H.-S. (2015). A survey of multi-agent formation control. *Automatica*, *53*(Supplement C), 424–440.

Pereira, A. R., & Hsu, L. (2008). Adaptive formation control using artificial potentials for euler-lagrange agents. *IFAC Proceedings Volumes*, *41*(2), 10788–10793.

Prorok, A., Correll, N., & Martinoli, A. (2011). Multi-level spatial modeling for stochastic distributed robotic systems. *The International Journal of Robotics Research*, *30*(5), 574–589.

Pugh, J., Raemy, X., Favre, C., Falconi, R., & Martinoli, A. (2009). A fast onboard relative positioning module for multirobot systems. *IEEE/ASME Transactions on Mechatronics*, *14*(2), 151–162.

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., et al. (2009) ROS: An open-source robot operating system. In *ICRA workshop on open source software* (Vol. 3, p. 5).

Rahmani, A., Ji, M., Mesbahi, M., & Egerstedt, M. (2009). Controllability of multi-agent systems from a graph-theoretic perspective. *SIAM Journal on Control and Optimization*, *48*(1), 162–186.

Roberts, J. F., Stirling, T., Zufferey, J. C., & Floreano, D. (2012). 3-D relative positioning sensor for indoor flying robots. *Autonomous Robots*, *33*(1), 5–20.

Roelofsen, S., Gillet, D., & Martinoli, A. (2015). Reciprocal collision avoidance for quadrotors using on-board visual detection. In *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 4810–4817). Washington: IEEE Press.

Rubenstein, M., Cornejo, A., & Nagpal, R. (2014). Programmable self-assembly in a thousand-robot swarm. *Science*, *345*(6198), 795–799.

Sapin, E. (2010). Gliders and glider guns discovery in cellular automata. In A. Adamatzky (Ed.), *Game of life cellular automata* (pp. 135–165). London: Springer.

Saska, M., Vonásek, V., Chudoba, J., Thomas, J., Loianno, G., & Kumar, V. (2016). Swarm distribution and deployment for cooperative surveillance by micro-aerial vehicles. *Journal of Intelligent & Robotic Systems*, *84*(1), 469–492.

Scheper, K. Y. W., & de Croon, G. C. H. E. (2016). Abstraction as a mechanism to cross the reality gap in evolutionary robotics. In E. Tuci, A. Giagkos, M. Wilson, & J. Hallam (Eds.), *From animals to animats 14: 14th international conference on simulation of adaptive behavior, SAB 2016, Aberystwyth, UK, August 23–26, 2016, Proceedings* (pp. 280–292). Cham: Springer.

Shiell, N., & Vardy, A. (2016). A bearing-only pattern formation algorithm for swarm robotics. In M. Dorigo, M. Birattari, X. Li, M. López-Ibáñez, K. Ohkura, C. Pinciroli, & T. Stützle (Eds.), *Swarm intelligence* (pp. 3–14). Cham: Springer.

Slavkov, I., Carrillo-Zapata, D., Carranza, N., Diego, X., Jansson, F., Kaandorp, J., et al. (2018). Morphogenesis in robot swarms. *Science Robotics*, *3*(25), eaau9178.

Smith, B., Howard, A., McNew, J.-M., Wang, J., & Egerstedt, M. (2009). Multi-robot deployment and coordination with embedded graph grammars. *Autonomous Robots*, *26*(1), 79–98.

Stegagno, P., Cognetti, M., Oriolo, G., Bülthoff, H. H., & Franchi, A. (2016). Ground and aerial mutual localization using anonymous relative-bearing measurements. *IEEE Transactions on Robotics*, *32*(5), 1133–1151.

Tanner, H. G. (2004). On the controllability of nearest neighbor interconnections. In *2004 43rd IEEE conference on decision and control (CDC)* (Vol. 3, pp. 2467–2472).
van der Helm, S., McGuire, K. N., Coppola, M., & de Croon, G. C. H. E. (2018). On-board range-based relative localization for micro aerial vehicles in indoor leader-follower flight. ArXiv Preprint. arXiv:1805.07171.
Van Steen, M. (2010). *Graph theory and complex networks: An introduction*. Amsterdam: Maarten van Steen.
Verhoeven, C. J. M., Bentum, M. J., Monna, G. L. E., Rotteveel, J., & Guo, J. (2011). On the origin of satellite swarms. *Acta Astronautica*, *68*(7–8), 1392–1395.
Werfel, J., & Nagpal, R. (2008). Three-dimensional construction with mobile robots and modular blocks. *International Journal of Robotics Research*, *27*(3–4), 463–479.
Werfel, J., Petersen, K., & Nagpal, R. (2014). Designing collective behavior in a termite-inspired robot construction team. *Science*, *343*(6172), 754–758.
Wessnitzer, J., Adamatzky, A., & Melhuish, C. (2001). Towards self-organising structure formations: A decentralized approach. In J. Kelemen & P. Sosík (Eds.), *Advances in Artificial Life* (pp. 573–581). Berlin: Springer.
Winfield, A. F., Sa, J., Fernández-Gago, M., Dixon, C., & Fisher, M. (2005). On formal specification of emergent behaviours in swarm robotic systems. *International Journal of Advanced Robotic Systems*, *2*(4), 39.
Winfield, A. F. T., & Nembrini, J. (2012). Emergent swarm morphology control of wireless networked mobile robots. In R. Doursat, H. Sayama, & O. Michel (Eds.), *Morphogenetic engineering: Toward programmable complex systems* (pp. 239–271). Berlin: Springer.
Winfield, A. F. T., Liu, W., Nembrini, J., & Martinoli, A. (2008). Modelling a wireless connected swarm of mobile robots. *Swarm Intelligence*, *2*(2), 241–266.
Yamauchi, Y., & Yamashita, M. (2013). Pattern formation by mobile robots with limited visibility. In T. Moscibroda & A. A. Rescigno (Eds.), *Structural information and communication complexity: 20th international colloquium, SIROCCO 2013, Ischia, Italy, July 1–3, 2013, Revised Selected Papers* (pp. 201–212). Cham: Springer.
Yamauchi, Y., & Yamashita, M. (2014). Randomized pattern formation algorithm for asynchronous oblivious mobile robots. In F. Kuhn (Ed.), *Distributed Computing* (pp. 137–151). Berlin: Springer.

Affiliations

Mario Coppola[1,2] · Jian Guo[2] · Eberhard Gill[2] · Guido C. H. E. de Croon[1]

Jian Guo
j.guo@tudelft.nl

Eberhard Gill
e.k.a.gill@tudelft.nl

Guido C. H. E. de Croon
g.c.h.e.decroon@tudelft.nl

1 Micro Air Vehicle Laboratory, Department of Control and Simulation, Faculty of Aerospace Engineering, Delft University of Technology, Kluyverweg 1, 2629 HS Delft, The Netherlands

2 Department of Space Systems Engineering, Faculty of Aerospace Engineering, Delft University of Technology, Kluyverweg 1, 2629 HS Delft, The Netherlands