The PageRank algorithm as a method to optimize swarm behavior through local analysis

Coppola, M.; Guo, J.; Gill, E.; de Croon, G. C.H.E.

DOI
10.1007/s11721-019-00172-z

Publication date
2019

Document Version
Final published version

Published in
Swarm Intelligence

Swarm Intelligence
https://doi.org/10.1007/s11721-019-00172-z

The PageRank algorithm as a method to optimize swarm behavior through local analysis

M. Coppola[1,2] · J. Guo[2] · E. Gill[2] · G. C. H. E. de Croon[1]

Received: 8 December 2018 / Accepted: 13 August 2019

Abstract
This work proposes PageRank as a tool to evaluate and optimize the global performance of a swarm based on the analysis of the local behavior of a single robot. PageRank is a graph centrality measure that assesses the importance of nodes based on how likely they are to be reached when traversing a graph. We relate this, using a microscopic model, to a random robot in a swarm that transitions through local states by executing local actions. The PageRank centrality then becomes a measure of how likely it is, given a local policy, for a robot in the swarm to visit each local state. This is used to optimize a stochastic policy such that the robot is most likely to reach the local states that are "desirable," based on the swarm's global goal. The optimization is performed by an evolutionary algorithm, whereby the fitness function maximizes the PageRank score of these local states. The calculation of the PageRank score only scales with the size of the local state space and demands much less computation than swarm simulations would. The approach is applied to a consensus task, a pattern formation task, and an aggregation task. For each task, when all robots in the swarm execute the evolved policy, the swarm significantly outperforms a swarm that uses the baseline policy. When compared to globally optimized policies, the final performance achieved by the swarm is also shown to be comparable. As this new approach is based on a local model, it natively produces controllers that are flexible and robust to global parameters such as the number of robots in the swarm, the environment, and the initial conditions. Furthermore, as the wall-clock time to evaluate the fitness function does not scale with the size of the swarm, it is possible to optimize for larger swarms at no additional computational expense.

Keywords Swarm robotics · Microscopic · Micro–macro link · Evolutionary algorithm · Pattern formation · Consensus · Aggregation · Local · PageRank · Centrality

1 Introduction

Machine learning techniques are a powerful approach to develop swarm behaviors. Evolutionary algorithms, for instance, can efficiently explore a solution space and extract viable local behaviors that fulfill a desired global goal (Nolfi 2002; Francesca and Birattari 2016).

✉ M. Coppola
m.coppola@tudelft.nl

Extended author information available on the last page of the article

Published online: 23 August 2019

They have been used on numerous architectures, including: neural networks (Izzo et al. 2014; Duarte et al. 2016), state machines (Francesca et al. 2015), behavior trees (Scheper et al. 2016; Jones et al. 2018), and grammar rules (Ferrante et al. 2013). A bottleneck of these algorithms is in the need to evaluate how the whole swarm performs against each controller that they generate. Because of the complexity of swarms and the difficulty in predicting the global outcome, a full simulation of the entire swarm is carried out each time. This is subject to scalability issues as the size of the swarm increases, for example:

1. The computational load required to execute the simulations increases with the size of the swarm;
2. It may take longer for the desired behavior to emerge, requiring a longer simulation time for each evaluation trial, especially in the initial stages of the evolution;
3. The evolved policy may be over-fitted to the global parameters used during the simulation, such as the number of robots, the initial conditions, or the environment;
4. A solution needs to be simulated multiple times in order to reliably assess the expected performance of a given behavior (Trianni et al. 2006). Avoiding re-evaluation may result in poor behaviors being erroneously assigned a higher fitness thanks to one lucky run, which may ultimately result in a performance drop (Di Mario et al. 2015a, b);[1]
5. The evolution may be subject to bootstrap issues (Silva et al. 2016; Gomes et al. 2013).

In this work, in order to tackle these scalability problems, we introduce a new approach to the field of swarm robotics: PageRank (Brin and Page 1998; Page et al. 1999). PageRank is a graph centrality and node ranking algorithm. It was originally developed by Sergey Brin and Larry Page as part of Google™. Its objective was to rank the importance of Webpages based on the hyperlink structure of the World Wide Web. PageRank's philosophy was to model the browsing behavior of a user who surfs the Web by randomly clicking through hyperlinks and to measure the value of Webpages based on how likely it would be for this user to visit them. In this paper, we port this idea to the world of swarm robotics. Here, a robot in a swarm becomes analogous to a Web surfer. The robot moves through local states by taking actions, much like a Web surfer navigates through Webpages by clicking hyperlinks. With PageRank centrality, we can then evaluate the relative likelihood with which the robot will end up in the local states. Then, with the knowledge that a desired global goal is more likely to be achieved when the robots are (or pass through) a given set of local states, we can efficiently quantify the global performance of the swarm in achieving its goal. More specifically, we propose a fitness function, based on PageRank, that can assess the *global* performance of a swarm while only evaluating the *local* model of a single robot in the swarm. This micro–macro link frees us from the need to simulate the swarm. Due to the local nature of this approach, the evaluation is independent from global parameters such as the size of the swarm, the initial condition, the environment, or lower-level controllers. The introduction of this method is the main contribution of this paper. We will showcase its potential by applying it to optimize the local behavior for three different swarming tasks: (1) consensus agreement, (2) pattern formation, and (3) aggregation.

We begin the paper by discussing related work in Sect. 2. Here, we place our contribution within the context of other solutions found in the literature which also had the aim of tackling scalability issues. We further compare our use of a PageRank-based microscopic model to other swarm modeling approaches. In Sect. 3, we then detail how PageRank works and explain how it can be applied to model, assess, and optimize the performance of a robotic

[1] An alternative to re-evaluation is to vary other parameters. For instance, one could simulate once but for a longer time (Di Mario and Martinoli 2014), although this is applicable to continuing task and not to tasks with a definite global goal.

swarm. The approach is then directly applied to optimize the behavior of three swarming tasks.

Consensus agreement (Sect. 4) In this task, we optimize the behavior of a swarm that must achieve consensus between multiple options. Each robot can sense the opinion of its neighbors and, based on a stochastic policy, decide whether it should change its opinion (and, if so, what to change its opinion to). Using PageRank, we optimize the stochastic policy so as to help the swarm achieve consensus as quickly as possible. The policy is optimized independently of the size of the swarm or its spatial configuration. We further optimize a more limited variant of this task whereby robots can not sense the opinion of their neighbors, but can only sense whether they are in agreement with their neighborhood or not.

Pattern formation (Sect. 5) For this task, we optimize the performance of a swarm of robots with the global goal of arranging into a desired spatial configuration. The robots in the swarm have very limited knowledge of their surroundings. This section extends our recent work published in Coppola and de Croon (2018), where we first attempted to optimize the behavior of a swarm in a pattern formation task, yet quickly encountered scalability problems for larger swarms/patterns. The scalability problems were a result of the fact that the swarm had to be simulated in order to assess the efficacy of a controller. This was infeasible for larger swarms, especially in early generations where performance is poor. Using the PageRank algorithm, we can now tackle these scalability issues, as it is no longer needed to simulate the swarm in order to assess the fitness of a behavior.

Aggregation (Sect. 6) In this task, we study a swarm of robots in a closed arena which should aggregate in groups of three or more. In comparison with the other two tasks, this optimization tunes a higher-level policy featuring two sub-behaviors (*random walk* and *stop*) with a probability that is dependent on the number of neighbors that a robot in the swarm can sense. The final achieved policy allows the swarm as a whole to aggregate successfully.

In Sect. 7, we discuss our general findings, including an analysis of the strengths and the current limitations of using PageRank centrality as a behavior optimization tool for swarm robotics. Section 8 provides concluding remarks.

2 Context and related work

In state of the art, the problems of scalability discussed in the introduction have mostly been tackled in two ways. First, there are methods that try to deal with the broad solution space that comes as the number of robots increases. For example, Gomes et al. (2012) used novelty search to encourage a broader exploration of the solution space. The second way is to use global insights to aid the evolutionary process. For example, Duarte et al. (2016) partitioned complex swarm behavior into simpler sub-behaviors. Hüttenrauch et al. (2017), with a focus on deep reinforcement learning, used global information to guide the learning process toward a solution. Alternatively, Trianni et al. (2006) and Ericksen et al. (2017) explored whether evolved behaviors for smaller swarms could generalize to larger swarms. In all cases, the need to simulate the swarm remains a bottleneck, for both evaluating the behavior and generalizing the behavior beyond the parameters used in simulation. In this work, we offer a different solution which discards simulation and exploits a micro–macro link based on evaluating the relative PageRank score between local states using only a local model of a single robot in the swarm. It extracts performance parameters without simulation

or propagation from an initial condition and only relies on an analysis of the model for a given policy.

This approach differs from other swarm modeling and evaluation solutions found in the literature, such as the multi-level modeling framework introduced by Lerman et al. (2001), Martinoli and Easton (2003), and Martinoli et al. (2004). There, the idea is to model the evolution of a swarm via probabilistic final state machines propagated in time. At the microscopic level, one probabilistic finite state machine is propagated from an initial condition for each robot in the swarm. At the macroscopic level, which can be readily abstracted from the microscopic model, the finite state machine describes the mean transition of robots between states. The macroscopic model probabilistically describes, at the global level, how many robots are in each state at a given point in time. This can also be expressed in terms of rate equations. For each level, the relevant transition rates between states are extrapolated from an analysis of the policy, as well as from geometric reasoning (e.g., based on the size of the arena and the expected robot density that ensues) (Martinoli et al. 2004) or from empirical data (Berman et al. 2007). To predict the global evolution of the swarm, the models are propagated in time from an initial condition, essentially simulating the swarm at an abstract level. For certain tasks, these models have been shown to be highly effective in predicting the general behavior of a swarm (Lerman et al. 2005). However, their accuracy and applicability are limited by the validity of the global assumptions which define the transition rates. For instance, to estimate the probability of interaction between any two robots, one assumption is that the system is "well mixed," meaning that the robots are all equally distributed within an area at all times and always have the same probability of encountering a neighbor. For reasons such as this, their use has been largely limited to the evaluation of tasks with few states and by swarms in bounded arenas. Examples where these models have been used are the modeling of collective decision making (Hamann et al. 2014; Reina et al. 2015), area exploration/foraging (Correll and Martinoli 2006; Campo and Dorigo 2007), or keeping an aggregate (Winfield et al. 2008).[2] For other tasks where the level of detail required is higher and/or where the global goal is achieved by a combination of several states, rather than all robots being in one state, this approach does not provide a sufficient level of detail. One such example is pattern formation, as treated in this paper, whereby the global goal is to achieve a particular spatial arrangement which cannot be described by such models.

Another class of macroscopic models, focused on spatial movements, models a swarm by using a diffusion model based on the Fokker–Planck equation (Hamann and Wörn 2008). This approach macroscopically models the general motion of a swarm under the assumption of Brownian motion. The Fokker–Planck equation provides an estimate for the density of robots in the environment as a result of the robots' motion. Prorok et al. (2011) explored the use of rate equations together with a diffusion model. This made it possible to study swarms in more complex environments where the swarm may not be well mixed, which otherwise causes drift errors (Correll and Martinoli 2006). However, there were still limitations on how to approximate more complex behaviors by the robots, which, by the nature of the assumption of Brownian motion, were limited to random walks.

The use of macroscopic models to optimize the behavior of robots in a swarm was explored by Berman et al. (2007, 2009, 2011) using models similar to those of the rate equations by Martinoli and Easton (2003). In these models, the fraction of robots in each state is modeled together with the rates at which the robots will transition between these states. It is then possible to optimize the transition rates such that the macroscopic model, on average, shows

[2] In the work by Winfield et al. (2008), the swarm operated in an unbounded arena. Instead, it was assumed that neighbors of each robot would be equally dispersed within the robot's sensing and communication range. This was shown to be reliable up to a certain extent, in part thanks to the behavior that was implemented.

that the swarm settles in certain states of interest. In (Berman et al. 2009) this is a done for the problem of task allocation, where it is also proven that all robots in the swarm will settle to a desired global equilibrium. However, as for rate equations, this approach is limited by the global level assumptions that are being taken in order to describe the mean of the swarm (Berman et al. 2011). Moreover, the outcome from such approaches can also be dependent on the initial condition of the swarm (Hsieh et al. 2008).

Macroscopic approaches thus make it possible to approximate how a swarm can evolve in time. They purposely refrain from incorporating the detailed experience of a single robot and rather approximate the mean evolution of the entire swarm. This makes them effective prediction tools, but they are limited in their ability to describe interactions to a higher level of detail. In contrast to multi-level models, the PageRank framework captures state transitions probabilistically, rather than temporally. We can apply this to analyze the impact of the actions by a single robot in a dynamic environment. This makes it able to tackle more specific tasks such as consensus agreement or pattern formation, with results that are found to be scalable, flexible, and robust to initial conditions or the number of robots.

3 PageRank centrality as a micro–macro link for swarm robotics

Centrality measures assess the relative importance of nodes in a graph based on the graph's topology. Several of these measures exist, which capture centrality from different perspectives. For example, *degree* centrality computes the importance of nodes based on the number of edges connected to them. Alternatively, *closeness* centrality measures the average shortest path between a node and all other nodes.[3] This paper deals with *PageRank* centrality. This is a graph centrality measure for directed graphs that measures the importance of each node recursively. This means that the PageRank centrality of a node is a function of the centrality of the nodes pointing to it, the centrality of the nodes pointing to those nodes, and so forth. This recursiveness indirectly accounts for the topology of the entire network, and it models how likely the node is to be reached when traversing the graph.

In this section, we detail how the PageRank centrality algorithm works (Sect. 3.1) and then explain how it can be used to microscopically model the possible state transitions of a robot in a swarm (Sect. 3.2). In Sect. 3.3, we then introduce a PageRank-based fitness function. This fitness function assesses, at a local level, a swarm's ability to achieve a desired global goal. It will be used to optimize the behavior for all tasks treated in this paper.

3.1 A review of PageRank centrality

Consider an arbitrary graph $G = (V, E)$ with nodes V and edges E. Let $u \in V$ be an arbitrary node in the graph. Following Page et al. (1999), a simplified expression for the PageRank $R(u)$ of a node $u \in V$ can be expressed as

$$R(u) = \sum_{v \in B_u} \frac{R(v)}{N_v}, \tag{1}$$

where B_u is the set of all nodes pointing to u, N_v is the number of outgoing edges of node v, and $R(v)$ is the PageRank of node v. Equation 1 serves to show the basic concept behind

[3] The interested reader is referred to the work of Fornito et al. (2016), where a summary and comparison of several centrality measures are provided.

PageRank: The PageRank of a node is a function of the PageRank of the nodes pointing to it. This means that being pointed to by a more important node will provide a node with a higher PageRank. In the case of the World Wide Web, for instance, this reflects how being linked to by a popular Webpage (whereby its popularity is also established in the same way) will then be evaluated as being more valuable than being linked to by a niche Webpage.

PageRank can be calculated simultaneously for all nodes using an iterative procedure. Here, we briefly review its key elements.[4] Let $\mathbf{R}$ be a vector that holds the PageRank of all nodes in V.

$$\mathbf{R}_{k+1}^{\intercal} = \mathbf{R}_{k}^{\intercal}\mathbf{G}. \tag{2}$$

$\mathbf{R}$ is obtained once Eq. 2 converges such that $|\mathbf{R}_{k+1}| - |\mathbf{R}_k| \leq \varepsilon$, where k is the iteration step and ε is a threshold (in this work we used $\varepsilon = 10^{-8}$). Equation 2 can be shown to converge quickly provided that the matrix $\mathbf{G}$ is stochastic and primitive (Langville and Meyer 2006).[5] $\mathbf{G}$ is known as the *"Google matrix."* In its full form, it is defined as

$$\mathbf{G} = \alpha(\mathbf{H} + \mathbf{D}) + (1 - \alpha)\mathbf{E}, \tag{3}$$

where:

- $\mathbf{H}$ is the adjacency matrix of graph G. $\mathbf{H}$ is a sub-stochastic transition probability matrix that describes the probability of transitioning between the nodes of graph G. For the example of the World Wide Web, $\mathbf{H}$ holds the hyperlink structure of the Web and it models how a Web surfer can navigate the Web by using hyperlinks.
- $\mathbf{D}$ is the *dangling node matrix*. The dangling nodes of graph G are nodes with no outgoing edges (e.g., no hyperlinks), which result in empty rows in $\mathbf{H}$. For the World Wide Web, if a user reaches a Webpage with no hyperlinks, then the user will resort to writing the name of a Webpage in the address bar. $\mathbf{D}$ describes these transitions and the probabilities thereof. The combined matrix $\mathbf{S} = \mathbf{H} + \mathbf{D}$ is a stochastic matrix.
- $\mathbf{E}$ is known as the *teleportation matrix*. This is an additional element that describes random transitions between nodes that are not captured by the topology of the graph G. For the World Wide Web, these transitions model how a Web surfer may choose, at any moment and on any Webpage, to manually type the address of a Webpage in the address bar instead of clicking through hyperlinks. In this case, the user will "teleport" to another Webpage regardless of whether a hyperlink to that Webpage exists. Note that matrices $\mathbf{D}$ and $\mathbf{E}$ have a non-null intersection. The information in $\mathbf{D}$ is included in $\mathbf{E}$. Both matrices describe transitions that occur as a result of a user typing the name of a Webpage in the address bar, except that $\mathbf{D}$ only holds those transitions for the cases where the user, having reached a dangling node, has no other option than to teleport.
- α is known as the *expansion factor*, where $0 \leq \alpha \leq 1$, which models the probability that a user follows hyperlinks as opposed to accessing a Webpage directly via the address bar. Note how Eq. 3 consists of two complementary terms. The first term models the transitions via hyperlinks (unless no hyperlinks exist as in the case of dangling nodes). The second term models "teleportation," described via $\mathbf{E}$. α describes the relative probability between

[4] For more details, including sample code and even some humorous anecdotes, we refer the reader to the book *"Google's PageRank and Beyond: The Science of Search Engine Rankings"* by Langville and Meyer (2006).

[5] A *stochastic matrix* holds the probability of transitioning between nodes as it would be described by Markov chains. It follows that the sum of each row must be equal to 1. A matrix $\mathbf{A}$ is *primitive* if $\exists k \forall (i, j) : A_{ij}^{k} > 0$. A primitive matrix is both irreducible and aperiodic. Irreducible means that any state in a Markov chain is reachable from any other state. Aperiodic means that there is no set period for returning to any given state. These properties allow the iterative algorithm to converge.

these two transition types. If $\alpha = 1$, the user always only follows hyperlinks (when they are available). If $\alpha = 0$, the user never follows hyperlinks and only teleports through the Web by typing Webpages in the address bar. Brin and Page (1998) advised to set $\alpha = 0.85$. Note that the simplified version of PageRank from Eq. 1 featured $\alpha = 1$.

In summary, the matrix **G** models how a user navigates the Web. **H** models the user's use of hyperlinks. **D** models what the user does when a Webpage with no hyperlinks is reached. **E** models the transitions that take place when the user chooses to not use hyperlinks, but directly go to a Webpage of choice. The matrices **H**, **D**, **E**, and the parameter α can then be tailored to a user's Web surfing behavior in order to produce more personalized results once the PageRank vector **R** is evaluated.

3.2 Using PageRank to model swarms

Just like Brin and Page used the Google matrix from Eq. 3 to model the behavior of a Web surfer, we can use it to model the behavior of a robot in the swarm. To do so, we must correlate the local experiences of a robot to a graph structure. We thus begin by defining the following discrete sets.

- Let $\mathcal{S}$ be the local state space of a robot. This is the set of local, discretized states that a robot can observe using its on-board sensors. The local states are going to be analogous to Webpages.
- Let $\mathcal{A}$ be the set of all discrete local actions that a robot can take.
- Let Π be a stochastic policy (a stochastic map between states $\mathcal{S}$ and actions $\mathcal{A}$) that the robot follows. This stochastic policy is analogous to the hyperlinks in the Web, as it allows robots to travel through local states.

Now consider a graph $G_{\mathcal{S}} = (V, E)$. The graph $G_{\mathcal{S}}$ is our microscopic local model of the robot in the swarm. The nodes of $G_{\mathcal{S}}$ are the local states that the robots can be in, such that $V = \mathcal{S}$. The edges of $G_{\mathcal{S}}$ are all transitions between local states that could take place. The local transitions can be of two types: *active* or *passive*. That is, either they can happen as a result of an action by the robot (active), or they can happen because the environment around the robot changes (passive). More formally, $G_{\mathcal{S}}$ is the union of two subgraphs: $G^a_{\mathcal{S}}$ and $G^p_{\mathcal{S}}$.

- $G^a_{\mathcal{S}}$, whereby the superscript a stands for "active," holds all state transitions that a robot could go through by an action of its own based on the stochastic policy Π. The edges of this graph are weighted based on the relative probabilities in the stochastic policy Π and thus represent how likely it is that the robot will take the action when in a given local state.
- $G^p_{\mathcal{S}}$, whereby the superscript p stands for "passive," holds all state transitions that a robot could go through because of changes in its environment, independently from Π.

The graphs $G^a_{\mathcal{S}}$ and $G^p_{\mathcal{S}}$ model the robot and the effects of the environment on the robot, respectively. These models are specific to the type of robot that is being used, its state space, its action space, and the nature of the task. For each of the three tasks that we explore in this manuscript (consensus, pattern formation, and aggregation), we will show how to define $G^a_{\mathcal{S}}$ and $G^p_{\mathcal{S}}$ accordingly. Once properly defined, the graphs $G^a_{\mathcal{S}}$ and $G^p_{\mathcal{S}}$ can then be expressed as the matrices **H**, **E**, and **D** introduced in Sect. 3.1

- **H** shall model how a robot navigates in its environment by its own actions, describing the possible local state transitions that may take place when the robot executes these actions. This is analogous to a user navigating through Webpages via hyperlinks. We thus define:

$$\mathbf{H} = \text{adj}(G^a_{\mathcal{S}}), \tag{4}$$

where adj(G) denotes the weighted adjacency matrix of a graph G.

– **E** models the "environment." These are state transitions that can happen to the robot even when it is not taking an action. For example, a neighbor could move away and cease to be a neighbor. This is analogous to a Web user who, instead of following hyperlinks through the Web, navigates to another Webpage via the address bar. In this case, the user instigates a state transition that is not described by the hyperlink structure (i.e., the "policy") of the World Wide Web. The matrix **E** is thus given by:

$$\mathbf{E} = \text{adj}(G^p_{\mathcal{S}}). \tag{5}$$

– **D** shall model what can happen to a robot as a result of the changing environment around the robot, *if and only if the robot reaches a state wherein it cannot (or will not) take any actions*. These states are states that the policy Π does not map to any actions. We group these states in the set $\mathcal{S}_{static} \subseteq \mathcal{S}$. Reaching such a state is analogous to reaching a Webpage with no hyperlinks. We define **D** as:

$$\mathbf{D} = \text{adj}(G^p_{\mathcal{S}}(S_{static})). \tag{6}$$

Note that the matrices **H**, **E**, and **D** must be row normalized in order for the iterative procedure of Eq. 2 to converge.

The expansion factor α remains to be defined, which models the probability that a robot will take an action over being subject to changes in the environment, as modeled by **E**. One may choose to keep α constant, as originally suggested by Page et al. (1999). For instance, α could be a constant value that is a function of the robot density. However, this would be a global parameter and would thus not be representative of the local experience of the robots, unlike the remainder of the model. Instead, we propose to make α a function of the local state that the robot is in. For instance, this may reflect that, if the robot is in a local state whereby it is surrounded by several neighbors, the probability that it undergoes passive transitions due to actions of its neighbors may be higher than if it is in a state whereby it is not surrounded by neighbors at all. While exploring the three tasks analyzed in this paper, we will define different approaches to express α in more detail.

3.3 Using PageRank to evaluate the performance of the swarm

Once a microscopic model of the swarm has been constructed using the framework described in Sect. 3.2, we can evaluate the PageRank centrality of all states in $\mathcal{S}$. This will inform us on how likely it is for a robot in the swarm to reach each local state. In turn, this will be used to evaluate the performance of the whole swarm.

This is where we devise the micro–macro link based on PageRank centrality. The true metric that we wish to optimize is the efficiency of the swarm in achieving a given global goal. However, directly assessing this parameter would require us to simulate the swarm (which may lead to the scalability problems introduced at the beginning of this paper), or use a macroscopic model of the swarm (which may fail to capture the details of the policy). Therefore, instead of directly optimizing the stochastic policy Π against the global performance metric, we optimize the likelihood that a robot in the swarm will end up in certain local states of interest, i.e., local states that we know should be reached for the global goal to happen. For example, in the aggregation task, we will wish to achieve (and stay in) local states with neighbors over local states without neighbors. Alternatively, in the consensus

task, the robot will aim to be in a state of agreement with its neighbors. Because all robots in the swarm are trying to achieve these "desired" local states as efficiently as possible, the global effect will then be that the swarm achieves the global goal more efficiently as well.

This concept can be formalized into a fitness function. Let $\mathcal{S}_{des} \subseteq \mathcal{S}$ be the set of local desired states pertaining to a global goal, and let $R(s)$ be the PageRank centrality of a state $s \in \mathcal{S}$. Based on this, we propose the following fitness function in order to evaluate the performance of the swarm:

$$F = \frac{\sum_{s \in \mathcal{S}_{des}} R(s)/|\mathcal{S}_{des}|}{\sum_{s \in \mathcal{S}} R(s)/|\mathcal{S}|}, \tag{7}$$

where $R(s)$ is the PageRank of state s. It is extracted following the calculation of $\mathbf{R}$ from Eq. 2 with the model of Sect. 3.2. This fitness function expresses the average PageRank of the states in $\mathcal{S}_{des}$ in relation to the average PageRank of all states $\mathcal{S}$. When used within an optimization strategy, our objective will be to alter Π so as to maximize F. This will maximize the average "popularity" of the states in $\mathcal{S}_{des}$ and increase the likelihood for the individual robot to be in one of the states. At the global level, we expect that if all robots act such that they are likely to enter a state $s \in \mathcal{S}_{des}$, then the final global goal will also be more likely to emerge. Defining the set of desired states for a given task may appear troublesome. However, throughout this paper we present three different tasks and show that the desired states can often be intuitively extracted from a global goal. We will return to this discussion in Sect. 7.

4 Sample task 1: consensus agreement

In this first task, a swarm of robots needs to achieve a consensus between a certain set of options. We will use the fitness function described in Sect. 3.3 to optimize the probability with which a robot should change its opinion such that the entire swarm achieves consensus as quickly as possible.

4.1 Task description and setting

Consider a set of N robots that must collectively choose between M options. As an example, these options could be possible sites to visit together with the rest of the swarm. We focus on the case where all options are of equal value, meaning that the robots do not have a preference for any particular option. Let $\mathcal{C}$ be the set of M options that the swarm considers. The global goal of the swarm is for all robots to settle on a choice $c \in \mathcal{C}$. We shall assume that the robots are in a static arbitrary connected configuration P.

The state s_i of a robot $\mathcal{R}_i$ holds the opinion of robot $\mathcal{R}_i$ and the number of neighbors with each of the other opinions. As an example, consider a swarm that must choose between $M = 2$ options, whereby $\mathcal{C} = \{A, B\}$. The local state space of a robot $\mathcal{R}_i$ is then described by all possible combinations of these three variables:

1. The internal opinion of $\mathcal{R}_i$, denoted c_i.
2. The number of neighbors of $\mathcal{R}_i$ with opinion A, denoted n_{iA}.
3. The number of neighbors of $\mathcal{R}_i$ with opinion B, denoted n_{iB}.

As each robot can only sense a maximum number of neighbors $N_{\max}$, there is a constraint $n_{iA}+n_{iB} \leq N_{\max}$. In reference to our setup, we shall set $N_{\max} = 8$. All possible combinations

of the above form the local state space $\mathcal{S}$ for this task. If $M > 2$, then the $\mathcal{S}$ can be expanded accordingly to accommodate all other relevant combinations.

Based on its local state s_i, a robot can choose to keep its opinion or change it. We set the action space $\mathcal{A}$ to $\mathcal{A} = \mathcal{C}$. Each robot follows a stochastic policy Π, dictating the probability of choosing an opinion $c \in \mathcal{C}$ (including its current one) for each state $s \in \mathcal{S}$. We then define the set $\mathcal{S}_{des}$ to be all states wherein the robot has the same opinion as all of its neighbors, for which we know a priori that there is no need to change opinion, as all robots are locally in agreement. These states are excluded from the stochastic policy Π, creating a global convergence point for the swarm. If all robots are in one of these local state, then it follows that consensus (which is the global goal) has been achieved. The size of the stochastic policy is $|\Pi| = (|\mathcal{S}| - |\mathcal{S}_{des}|) * |\mathcal{A}|$.

4.2 PageRank model

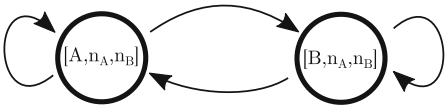
To define the PageRank model, we need to define $G^a_{\mathcal{S}}$, which denotes the state transitions whenever a given robot takes an action, and $G^p_{\mathcal{S}}$, which models the state transitions whenever the neighbors of the robot take actions.

Let us begin by defining $G^a_{\mathcal{S}}$. In this case, whenever a robot $\mathcal{R}_i$ changes its opinion between the M options that are available, only its internal choice changes, while the choice that its neighbors hold is not (directly) impacted by the action. Thus, for the example of $M = 2$, the only parameter of its state that changes is c_i, while n_{iA} and n_{iB} remain constant. $G^a_{\mathcal{S}}$ is thus formed by a set of disconnected subgraphs, each where the robot is only capable of changing its own internal state, but all other parameters stay constant. A sample subgraph of $G^a_{\mathcal{S}}$ for $M = 2$ is depicted in Fig. 1a.

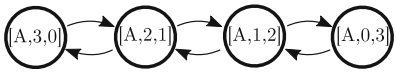
$G^p_{\mathcal{S}}$ follows the same philosophy as $G^a_{\mathcal{S}}$, but from the opposite perspective. The internal choice c_i of a robot $\mathcal{R}_i$ does not change when its neighbors change opinions. What changes is the number of neighbors with a certain opinion. The model assumes that the robot will always notice every time one of its neighbors changes opinion. $G^p_{\mathcal{S}}$ thus also takes the form of several disconnected subgraphs. An example of a subgraph for $M = 2$ is depicted in Fig. 1b for a robot with three neighbors. The matrices $\mathbf{H}$, $\mathbf{E}$, and $\mathbf{D}$ are extracted from $G^a_{\mathcal{S}}$ and $G^p_{\mathcal{S}}$ using Eqs. 4, 5, and 6, respectively.

Finally, we define the parameter α. For this task, we can reason that, if the robot is surrounded by more neighbors, then the likelihood that it will change opinion before one of its neighbors decreases. We can include this in our local model via a re-definition of α. To do so, we define a vector $\boldsymbol{\alpha}_{\mathbf{v}}$ that holds a different value of α for each state $s \in \mathcal{S}$, such that the equation for the Google matrix is modified to:

$$\mathbf{G} = \mathbf{D}_{\alpha_v} (\mathbf{H} + \mathbf{D}) + \left(\mathbf{I} - \mathbf{D}_{\alpha_v}\right) \mathbf{E}, \tag{8}$$



(a) Depiction of a subgraph of $G^a_{\mathcal{S}}$ for a robot which can change its opinion from A to B and vice versa. Its neighbors do not change opinion, hence n_A and n_B remain constant.



(b) Depiction of a subgraph of $G^p_{\mathcal{S}}$ for a robot with opinion A and 3 neighbors. All transitions have equal probability.

Fig. 1 Representation of subgraphs of $G^a_{\mathcal{S}}$ and $G^p_{\mathcal{S}}$ for a consensus task with $M = 2$ options, $C = \{A, B\}$

where $\mathbf{D}_{\alpha_v}$ is a diagonal matrix holding the vector $\boldsymbol{\alpha}_\mathbf{v}$. The entries of $\boldsymbol{\alpha}_\mathbf{v}$ are $\alpha_i \leq 1$ for $i = 1, \ldots, |\mathcal{S}|$. In this work, we model α_i for a given state $s_i \in \mathcal{S}$ using the following general definition:

$$\alpha_i = p_{action}(s_i) * \frac{1}{n_{neighbors}(s_i) + 1}, \tag{9}$$

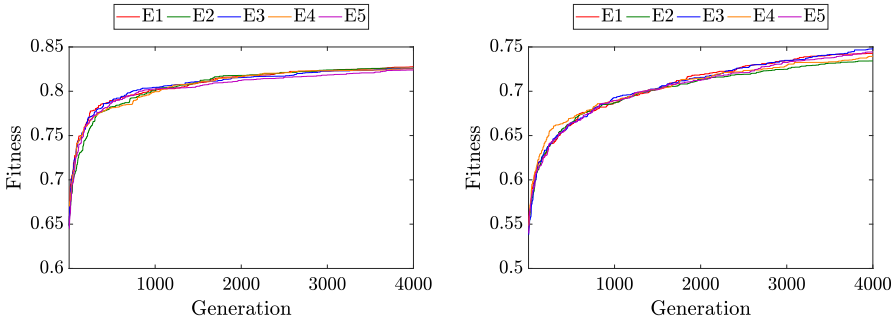
where $p_{action}(s_i) = \sum_{a \in \mathcal{A}} P(a|s_i)$. Thus, $p_{action}(s_i)$ is the cumulative probability of taking an action (any action) from the stochastic policy Π when in state s_i. All states $s_i \in \mathcal{S}_{active}$ feature $0 \leq p_{action}(s_i) \leq 1$. If $p_{action}(s_i) < 1$, then there is also a probability that, when in state s_i, the robot will not take an action but remain idle. The parameter $n_{neighbors}(s_i)$ is the number of neighbors at state s_i. As this parameter increases, α_i decreases. This same definition α will also be used for the pattern formation task in Sect. 5.

With the above, we have now fully defined a microscopic model of the swarm from the perspective of an individual robot while using PageRank's framework. We now proceed to optimize Π in order to maximize the fitness function expressed in Eq. 7.

4.3 Genetic algorithm setup and results

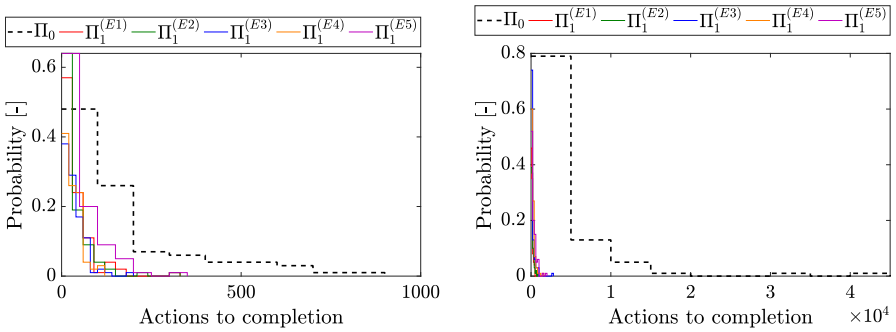
The optimization in this work is done via a genetic algorithm (GA). The GA features a population of ten scalar genomes, where the size of each genome is equal to Π. Each gene in a genome holds a value $0 \leq p \leq 1$, indicating the probability of executing the corresponding state–action pair from Π. Each new generation is produced by elite members (30%), offspring (40%), and mutated members (30%). Offspring are generated from averaging two parents' genomes. Mutation replaces 10% of a genome's genes with random values from a uniform distribution. The initial generation was created by assigning random values between 0 and 1 to all genes in each genome, following a uniform distribution. Note that the cumulative probability of taking any of the actions when in a given state is always normalized, since all choices are part of the policy.

With this setup, we evolved behaviors for the case where $M = 2$ and $M = 3$. The results of the fitness increase over five evolutionary runs are shown in Fig. 2, where we show the best genome that was evolved for both the case where $M = 2$ (Fig. 2a) and the case where $M = 3$ (Fig. 2b).



(a) $M = 2$ **(b)** $M = 3$

Fig. 2 Evolution of policy for consensus agreement task with two choices ($M = 2$) and three choices ($M = 3$)



(a) Swarm of 10 robots **(b)** Swarm of 20 robots

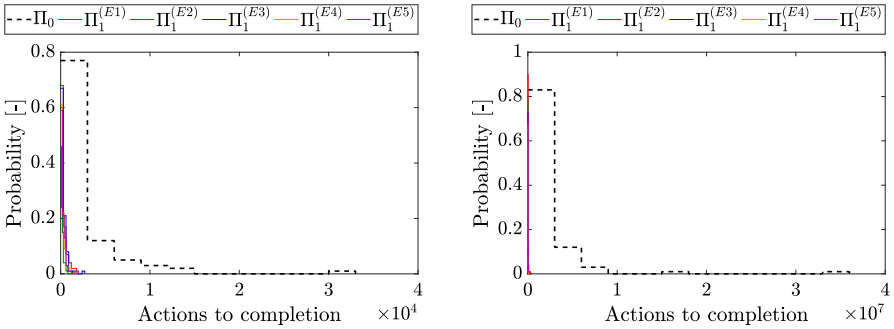
Fig. 3 Performance of consensus agreement task where $M = 2$, meaning that the robots must settle between two choices. The global performance is measured by the number of times that, cumulatively, the robots change their decision before a consensus is achieved

We now test the results of the evolved behavior to examine the magnitude by which the global performance of the swarm has improved. To do so, the swarm is simulated in discrete time whereby, at each time step, a random robot in the swarm changes its opinion. This models random concurrency in the swarm. For simplicity, we assume that the robots are placed on a grid world. Each robot $\mathcal{R}_i$ is capable of sensing the opinion of any robot $\mathcal{R}_j$ that happens to be in the eight grid points that surround it. We measure the performance of the swarm as the cumulative number of times that the robots in the swarm take an "action" before consensus is achieved, whereby an "action" is whenever a robot renews its opinion. This is indicative of how quickly the swarm is capable of reaching a consensus to one of the options. The results are compared to a basic baseline behavior where the robots choose between options with equal probability (except for the states $\mathcal{S}_{des}$, in which case they are in full agreement with their neighborhood and remain idle). In the following, the baseline behavior is denoted Π_0 and the evolved behaviors are denoted Π_1.

The tests were performed for swarms of ten robots and 20 robots. Each swarm was evaluated 100 times, in random configurations and from random starting conditions. The results for a swarm that must choose between $M = 2$ options are shown in Fig. 3. The results for a swarm that must choose between $M = 3$ options are shown in Fig. 4. In all cases, the swarm is capable of achieving a consensus substantially faster than with the baseline behavior. Moreover, we see that the performance is robust to the number of robots in the swarm. These results show that the evolutionary procedure was capable of finding a local behavior that provides an efficient outcome at the global level and also adapts well to the number of robots, the initial conditions, and the spatial configuration of the robots.

4.4 Variant with limited binary cognition

To gain further insight into the adaptability of the framework, we consider a limited variant of the consensus task where robots are endowed with binary sensors. Each robot is only capable of sensing whether *all* of its neighbors agree with it or not, but is unable to discern the opinions of individual neighbors. Such a case reflects the impact of high noise or poor sensing abilities on the part of the robots.



(a) Swarm of 10 robots **(b)** Swarm of 20 robots

Fig. 4 Performance of consensus agreement task where $M = 3$, meaning that the robots must settle between three choices. The global performance is measured by the number of times that, cumulatively, the robots change their decision before a consensus is achieved

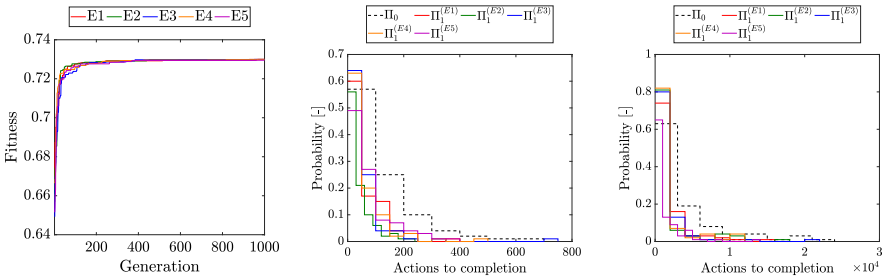
Consider the case where robots must choose between $M = 2$ options, with choices $\mathcal{C} = \{A, B\}$. It follows that each robot in the swarm can be in one of four local states:

s_{A0}: Internal opinion A, but the robot is not in agreement with all neighbors.
s_{A1}: Internal opinion A, and the robot is in agreement with all neighbors.
s_{B0}: Internal opinion B, but the robot is not in agreement with all neighbors.
s_{B1}: Internal opinion B, and the robot is in agreement with all neighbors.

The local state space is $\mathcal{S} = \{s_{A0}, s_{A1}, s_{B0}, s_{B1}\}$, and $\mathcal{S}_{static} = \mathcal{S}_{des} = \{s_{A1}, s_{B1}\}$. If in a state $s \notin \mathcal{S}_{des}$, a robot can choose between two actions.

a_A: Select opinion A
a_B: Select opinion B

The policy is evolved following the same evolutionary setup as in Sect. 4.3. The only difference is that, as the robots are now incapable of sensing the number of robots in their neighborhood, we set $\alpha = 0.3$ for all states, instead of using Eq. 9. The results of five evolutionary runs are shown in Fig. 5a. As expected, all evolutionary runs evolve to similar



(a) The best fitness over five evolutionary runs

(b) Performance improvements with a swarm of 10 robots of the evolved policies versus the baseline policy

(c) Performance improvements with a swarm of 20 robots of the evolved policies versus the baseline policy

Fig. 5 Evolution and performance of the policy for the binary variant of the consensus task. The baseline policy is given in Table 1a, and the evolved policies, for all cases, are similar to the one given in Table 1b

Table 1 Baseline policy and evolved policy for the binary variant of the consensus task for the case of $M = 2$

Π_0	a_A	a_B
(a) Baseline policy Π_0		
s_{A0}	0.5	0.5
s_{B0}	0.5	0.5
Π_1	a_A	a_B
(b) Evolved unbiased policy Π_1		
s_{A0}	0.002	0.998
s_{B0}	0.999	0.001

In the original policy (denoted Π_0), the robots have equal probability of selecting either choice. In the evolved policy (denoted Π_1), the robots always switch their opinion whenever their neighborhood is not in agreement with their choice. For Π_1, the reason that the values are not exactly 1 and 0 is attributed to the mutation strategy that was used during the evolution

results. The performance improvement over a baseline case (whereby the robots alternate between states with equal probability) is shown in Fig. 5b, c for swarms of ten robots and 20 robots, respectively.

4.5 Analysis

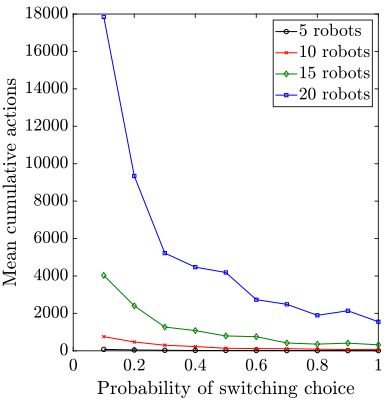
The small size of the solution space for the variant of the consensus task studied in Sect. 4.4 provides an opportunity to analyze the evolved policy and the global fitness landscape in detail. The baseline policy and the evolved policies are given in Table 1a, b, respectively. In Table 1b, it can be seen that our evolutionary procedure determined that the best strategy would be for the robots to, almost deterministically, switch their opinion.[6] We investigated whether this was an optimum solution by simulating the system for different probabilities of switching opinion. This was done using the same simulation setup as all evaluations in this section, whereby each setting was evaluated and averaged over 100 runs.

A first set of results for these simulations is shown in Fig. 6a, which shows the performance of the swarm in relation to the probability of switching. Here, it is confirmed that the policy is correct. As the probability of switching choices increases, the swarm is capable of achieving consensus more efficiently. It can also be seen that the evolved policy scales well with the size of the swarm.

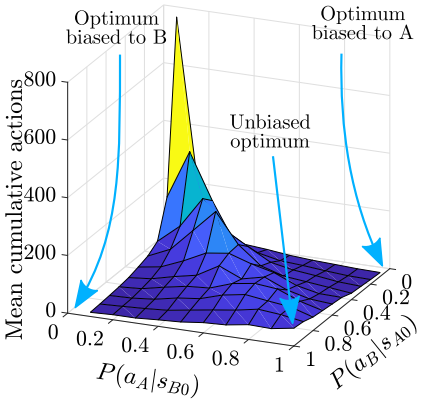
We further investigated the entire solution space for a swarm of ten robots. Here, it is revealed that the evolved policy Π_1, as given in Table 1b, is not the global optimum, but the unbiased global optimum. By unbiased, we mean that the robots are equally probable of selecting option A as they are of selecting option B. However, more optimal options exist if the robots become biased toward one of the two options. This result is actually reflected in the PageRank-based fitness function with which the behavior was evolved, as we had provided the set $\mathcal{S}_{des} = \{s_{A1}, s_{B1}\}$ and gave equal importance to both options. It then follows that this brought us to the unbiased optimum.

To test whether our framework could adapt to this, we also evolved the policy for the cases where $\mathcal{S}_{des} = \{s_{A1}\}$ and where $\mathcal{S}_{des} = \{s_{B1}\}$. Note that we kept $\mathcal{S}_{static} = \{s_{A1}, s_{B1}\}$, because

[6] We attribute the fact that the solution is nearly deterministic, and not fully deterministic, to our GA implementation.



(a) Correlation between the global performance of the swarm and the probability of switching opinion for swarms of different sizes.



(b) Correlation between the global performance of the swarm and the probability of switching opinion to choices A or B for a swarm of 10 robots.

Fig. 6 Depiction of global policy analysis for the probability of switching between two options A and B in the limited variant of the consensus task

we still want the robots to stop if they are in agreement with their neighborhood. The results of these two evolutions are given in Table 2a, b for $\mathcal{S}_{des} = \{s_{A1}\}$ and $\mathcal{S}_{des} = \{s_{B1}\}$, respectively. The evolution discovered the biased optimum for both options. This shows a flexibility on the part of the algorithm to adapt to the desired goals. Originally, the algorithm had evolved the unbiased policy because it had been told that both options had equal importance. When this constraint was altered to a new desired goal, the algorithm evolved a policy which reflected the new goal.

Additionally to the analysis above, the scalability of the evolved policies was analyzed in order to better understand their performance as swarm size increases. The comparison results showing the average performance of the policies are shown in Fig. 7a, b. This comparison is based on the results shown in the previous sections. It can be seen that the evolved policies scale well with the size of the swarm. The policy that was evolved with knowledge of the neighborhood scales gracefully, with only a minimal increase in the number of actions taken (per agent) as swarm size increases. A reason for this increase is the fact that, in a larger swarm, a robot is more likely to be surrounded by more neighbors, thus experiencing more uncertainty in its neighborhood. Furthermore, note that when a robot chooses to stay with its current choice, then this is also counted as an action. As expected, the limited variant evolved in Sect. 4.4, by nature of the fact that the robots have less data on their local surroundings, begins to struggle more as the swarm size increases.

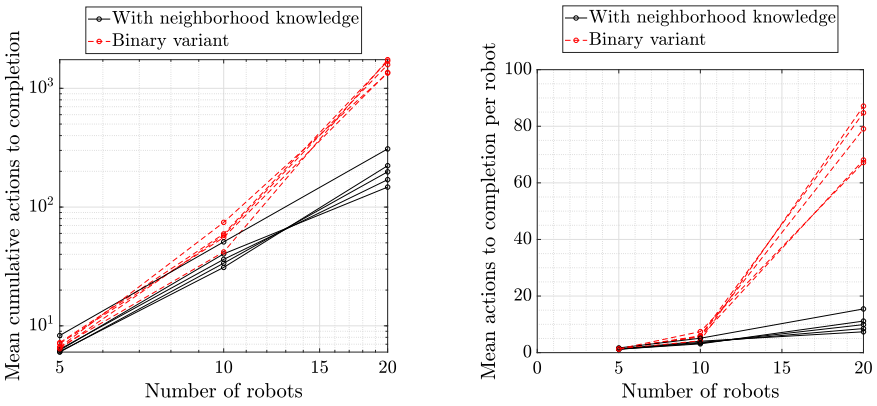
5 Sample task 2: pattern formation

This task deals with a swarm of robots with low awareness of their surroundings that must arrange into a desired spatial configuration. The robots are homogeneous (identical and without hierarchy), anonymous (do not have identities), reactive (memoryless), cannot communicate, do not have global position information, do not know the goal of the swarm, and operate asynchronously (i.e., by local clocks) in an unbounded space. The only knowledge

Table 2 Evolved policies with a bias toward options A or B

Π_{bA}	a_A	a_B
(a) Policy Π_{bA} with bias to option A		
s_{A0}	0.9982	0.0018
s_{B0}	0.9982	0.0018
Π_{bB}	a_A	a_B
(b) Policy Π_{bB} with bias to option B		
s_{A0}	0.0004	0.9996
s_{B0}	0.0042	0.9958

For both policies, the reason that the values are not exactly 1 and 0 is attributed to the mutation strategy that was used during the evolution



(a) Mean cumulative actions taken by the swarm **(b)** Mean number of actions taken per robot

Fig. 7 Comparison of mean performance of the evolved policies for both variants of the consensus task for the case where $M = 2$

available to the robots is: (1) a common heading direction, such as north, and (2) the relative location of their neighbors within a maximum range. In previous work (Coppola et al. 2019), we have developed a local behavior with which such limited robots can always eventually arrange into a global desired pattern. Moreover, we provided a set of conditions to verify whether a given behavior would always eventually lead to the emergence of the desired pattern.

One issue that remained to be solved was that even simple patterns were found to take up to hundreds of actions (cumulatively by all robots) to emerge, and this number appeared to grow with the complexity of the pattern and the size of the swarm. Although this is to be expected, in light of the limited knowledge on the part of the robots, it is an issue that cannot be ignored if the system is to be used on real robots with limited battery life and other real-world time constraints. Solving this via classical approaches, wherein the swarm had to be ultimately simulated in order to find an optimum policy, proved unscalable (Coppola and de Croon 2018). This was because of all the issues listed in the introduction of this paper. In this section, we show how we can circumvent the scalability problem by tackling the optimization using the PageRank algorithm.

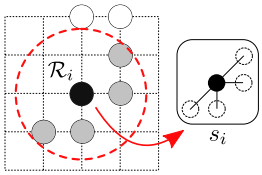
5.1 Description of approach to pattern formation

Such that this manuscript may be self-contained, this section summarizes the pattern formation methodology from Coppola et al. (2019), wherein a more detailed explanation can be found. For the sake of brevity, in this work we will assume that the swarm operates in a grid world and in discrete time. However, as demonstrated in Coppola et al. (2019), the behavior can also be used in continuous time and space with robots operating with local clocks.

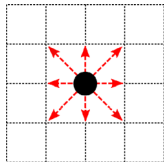
5.1.1 Setting

Consider N robots that exist in an unbounded discrete grid world and operate in discrete time. In the case studied in this paper, each robot $\mathcal{R}_i$ can sense the location of its neighbors in the eight grid points that surround it, as depicted in Fig. 8a. This is the local state s_i of the robot, which is all the information that it has. The local state space $\mathcal{S}$ consists of all combinations of neighbors that it could sense, such that $|\mathcal{S}| = 2^8$. At time step $k = 0$, we assume the swarm begins in a connected topology forming an arbitrary pattern P_0. At each time step, one random robot in the swarm takes an action, whereby it is able to move to any of the 8 grid points surrounding it, as depicted in Fig. 8b. This is the action space of the robots, denoted $\mathcal{A}$. If a robot takes an action at one time step, then it will not take an action at the next time step. This models the fact that a real robot would need some time to settle after each action and reassess its situation, leaving a time window for its neighbors to move.

The goal of the swarm is to rearrange from its initial arbitrary pattern P_0 into a desired pattern P_{des}. This is achieved using the following principle. The local states that the robots are in when P_{des} is formed are extracted, and form a set of local desired states $\mathcal{S}_{des} \subseteq \mathcal{S}$, as depicted by the examples in Fig. 9. These local desired states are extracted as the observations of the robots once the pattern is formed, similarly to how puzzle pieces form a puzzle. If robot $\mathcal{R}_i$ finds itself in any state $s_i \in \mathcal{S}_{des}$, then it is instructed to not move, because, from its perspective, the goal has been achieved. Given a P_{des} and the corresponding $\mathcal{S}_{des}$, it can be automatically (and sometimes even intuitively) verified whether the local desired states will uniquely form P_{des}, or whether they can also can give rise to spurious global patterns. If spurious global patterns are not possible, then, until P_{des} is formed, at least one robot will be in a state $s \notin \mathcal{S}_{des}$ and will seek to amend the situation. The swarm will then keep reshuffling unless P_{des} forms (Coppola et al. 2019).



(a) Example of a local state $s_i \in \mathcal{S}$ of a robot $\mathcal{R}_i$. The robot is shown in black and its neighbors within sensing range are shown in gray. It is assumed that a robot can always sense neighbors in the 8 grid points that surround it.



(b) Possible actions that a robot can take. The robot is capable of moving omni-directionally to any of the 8 grid points around its current position. These 8 actions form the actions space $\mathcal{A}$.

Fig. 8 Depictions of a local state and the actions that a robot in the swarm can take

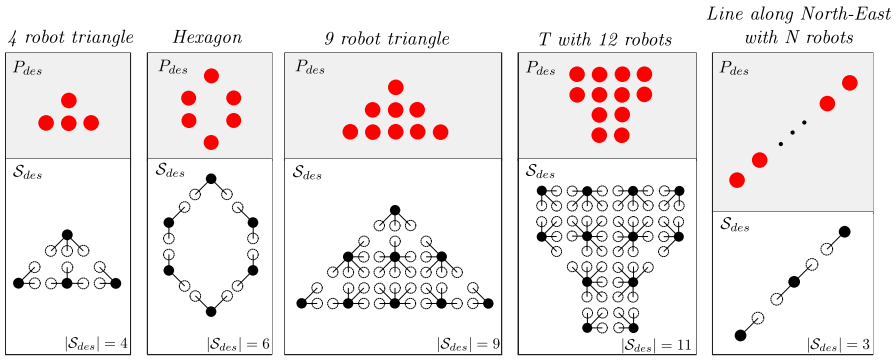


Fig. 9 Set of desired states $\mathcal{S}_{des}$ for the exemplary patterns treated in this paper, featuring patterns of increasing complexity and/or size. The set $\mathcal{S}_{des}$ can be intuitively extracted for each pattern as the "puzzle pieces" that compose it. Note that the T with 12 robots has 11 states in $\mathcal{S}_{des}$. This is because the top state repeats. Also note that the line always only has three states in $\mathcal{S}_{des}$, because the center state can repeat indefinitely

5.1.2 Baseline behavior of the robots

When a robot $\mathcal{R}_i$ is in a state $s_i \notin \mathcal{S}_{des}$, it should execute an action from $\mathcal{A}$. From the state space and action space, we can formulate a stochastic policy $\Pi = \mathcal{S} \times \mathcal{A}$. However, not all actions should be allowed. The actions that: a) cause collisions and b) cause local separation of the swarm are eliminated from Π, because they are not "safe." Moreover, as explained in Sect. 5.1.1, all states $s \in \mathcal{S}_{des}$ do not take any actions. From this, we extract a final stochastic policy Π_0, where $\Pi_0 \subseteq \Pi$. When a robot is in a state s, it will use the policy Π_0 to randomly select one possible action from the available state–action pairings. In Coppola et al. (2019), the final action was chosen from the available options based on a uniform probability distribution.

It is important to note that, from this pruning process, there also emerge additional local states that cannot take any actions. A robot in such a state will not be able to move or else it will either collide with other robots or possibly cause separation of the swarm. We refer to such states as *blocked* states. The set of blocked states is denoted $\mathcal{S}_{blocked}$. The states in $\mathcal{S}_{des}$ and $\mathcal{S}_{blocked}$ are functionally equivalent. In either case, a robot will not take any action. Together, they form the umbrella set $\mathcal{S}_{static} = \mathcal{S}_{des} \cup \mathcal{S}_{blocked}$.

Furthermore, conceptually in contrast to static states, there are states where a robot will be capable of moving away from and/or around its neighborhood without issues. We call these states *simplicial*. Simplicial states are characterized by only having one *clique*, where we define a clique as a connected set of neighbors.[7] The set of simplicial states is denoted $\mathcal{S}_{simplicial}$. Figure 10 shows examples of blocked states (Fig. 10a, b), a simplicial state (Fig. 10c), and a non-simplicial state (Fig. 10d).

5.1.3 Modeling the local experiences of the robots

Let us now construct the graph $G_{\mathcal{S}} = (V, E)$ that models the local experiences of the robots. The nodes of $G_{\mathcal{S}}$ are the local states that the robots can be in, such that $V = \mathcal{S}$. The edges of $G_{\mathcal{S}}$ are all possible transitions between local states.

[7] If the neighbors of a robot form only one clique, then this means that, if this robot were to disappear, it is guaranteed that its neighbors would all remain connected among each other.

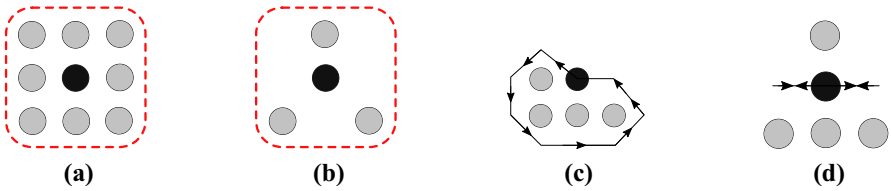


Fig. 10 Examples of: **a** a state $s \in \mathcal{S}_{blocked}$, due to it being surrounded; **b** a state $s \in \mathcal{S}_{blocked}$, because any motion will cause the swarm to locally disconnect; **c** a state $s \in \mathcal{S}_{active} \cap \mathcal{S}_{simplicial}$, because it can travel around all its neighbors; **d** a state $s \in \mathcal{S}_{active}$ but $s \notin \mathcal{S}_{simplicial}$, because it can move but it cannot travel around all its neighbors or else it might disconnect the swarm

For this task, we break down $G_{\mathcal{S}}$ in three subgraphs.

- $G^1_{\mathcal{S}}$ indicates all state transitions that a robot could go through by an action of its own, based on Π_0.
- $G^2_{\mathcal{S}}$ indicates all state transitions that a robot could go through by an action of its neighbors, which could also move out of view.
- $G^3_{\mathcal{S}}$ indicates all state transitions that a robot could go through if another robot, previously out of view, were to move into view and become a new neighbor.

Additionally, let $G^{2r}_{\mathcal{S}}$ be a subgraph of $G^2_{\mathcal{S}}$. $G^{2r}_{\mathcal{S}}$ only holds the state transitions in $G^2_{\mathcal{S}}$ where neighbors stay in view, and does not hold the ones where neighbors fall out of view.

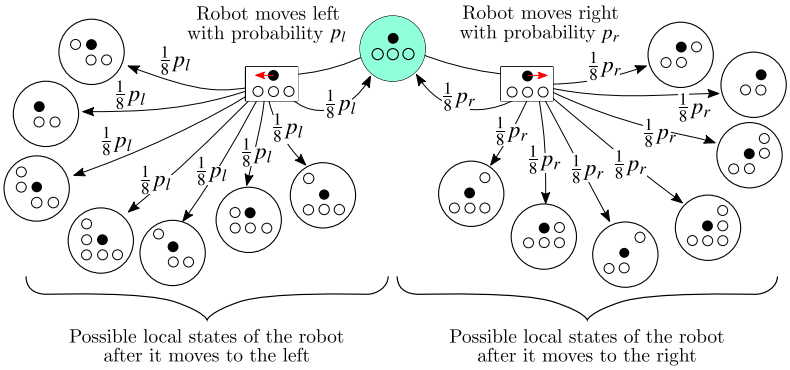
To help visualize $G^1_{\mathcal{S}}$, $G^2_{\mathcal{S}}$, and $G^3_{\mathcal{S}}$, Fig. 11 shows a node and its successors for each graph. Figure 11a shows a node and its successors for $G^1_{\mathcal{S}}$ following an action based on a sample policy Π_0. Figure 11b, c shows the same node in graphs $G^2_{\mathcal{S}}$ and $G^3_{\mathcal{S}}$, respectively. As it can be seen, these three graphs bear a strong resemblance to the graphs $G^a_{\mathcal{S}}$ and $G^p_{\mathcal{S}}$ needed to define PageRank. We shall return to this in Sect. 5.2, where the microscopic PageRank model for this task is defined.

5.1.4 Verifying that the pattern will always eventually emerge

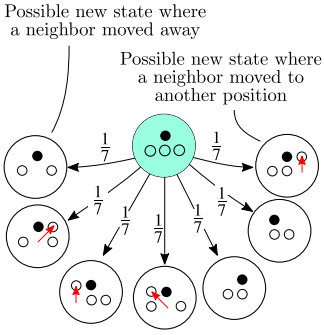
In Coppola et al. (2019), it was shown that, by analyzing certain properties of $G^1_{\mathcal{S}}$, $G^2_{\mathcal{S}}$, and $G^3_{\mathcal{S}}$, it can be verified that the pattern P_{des} will eventually form starting from any initial pattern P_0. This way, we can assess whether a policy is such that the final pattern will always eventually emerge, or whether it may cause the swarm to reshuffle endlessly without ever settling into the pattern. (We refer to this situation as a livelock.)

The conditions are repeated here because, once we optimize the policy using PageRank, they shall be used as constraints in order to ensure that the final policy always eventually achieves the pattern. Specifically, the following conditions need to be met:

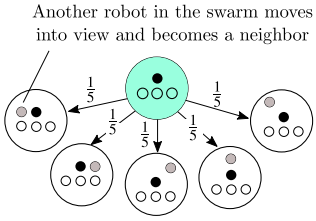
1. $G^1_{\mathcal{S}} \cup G^2_{\mathcal{S}}$ shows that each state in $\mathcal{S}$ features a path to each state in $\mathcal{S}_{des}$.
2. For all states $s \in \mathcal{S}_{static} \cap \mathcal{S}_{\neg simplicial} - s_{surrounded}$, none of the cliques of each state can be formed only by robots that are in a state $s \in \mathcal{S}_{des} \cap \mathcal{S}_{simplicial}$. Here, $s_{surrounded}$ is the state that is surrounded by neighbors along all directions.
3. $G^{2r}_{\mathcal{S}}$ shows that all static states with two neighbors can directly transition to an active state.
4. $G^1_{\mathcal{S}}$ shows that any robot in state $s \in \mathcal{S}_{active} \cap \mathcal{S}_{simplicial}$ can travel around all its local neighbors, as exemplified in Fig. 10c (with the exception of when a loop is formed or when it enters a state $s \in \mathcal{S}_{static}$).



(a) Example of a state (i.e., node) $s \in \mathcal{S}$ in graph $G^1_{\mathcal{S}}$ and its successor nodes. The successor nodes are extracted based on the possible actions to which the state s is mapped in Π_0. As the robot moves, it may end up in several states (including its original state) depending on the type of neighborhood that it finds after its action. Because it cannot see ahead, all options are possible until the action is executed. Therefore, all successor nodes following an action can be reached with equal probability. For this example, p_l indicates the probability of taking an action to the left, and p_r indicates the probability of taking an action to the right, according to the policy, when in the state s.



(b) Example of a state (node) in graph $G^2_{\mathcal{S}}$ and its successor nodes. These are all possible states that the robot may be in after one of its neighbors takes an action. Because the robot does not know what states its neighbors are in, it is assumed that neighbors can take any action from $\mathcal{A}$ with equal probability. All state transitions have the same probability.



(c) Example of the state (node) in graph $G^3_{\mathcal{S}}$ and its successors. State transitions are caused by a new neighbor arriving at an empty grid point. All state transitions have the same probability of being reached.

Fig. 11 Examples of a node and its successor nodes from graphs $G^1_{\mathcal{S}}$, $G^2_{\mathcal{S}}$, and $G^3_{\mathcal{S}}$

5. In $G^3_{\mathcal{S}}$, 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).

The detailed motivations behind these conditions, published in Coppola et al. (2019), are not repeated here due to page constraints. In summary, they ensure that all robots will keep moving around with sufficient freedom for the swarm to reshuffle until the pattern is achieved. Condition 1 checks that it is possible to reach the final local desired states independently of the initial local states. Conditions 2 and 3 check that there is always at least one robot in the swarm that has the potential to move with sufficient freedom. Conditions 4 and 5 check that the free robot(s) is (or are) capable of sufficient exploration.

These conditions are local in nature. They analyze the local states of a robot based on its limited sensing range and the actions that the robot could take as a result. The advantage of this is that checking whether the conditions are met is independent of the size of the swarm, avoiding the combinatorial explosion that would otherwise ensue. Note that the conditions are *sufficient*, but not *necessary*. Fulfilling them means that the pattern will be achieved, but not fulfilling them does not mean the pattern will not be achieved.[8] The first four patterns in Fig. 9 pass the proof conditions. The line, instead, does not fulfill all conditions. Moreover, it can be subject to spurious patterns (for example, a slanted H) as a result of $\mathcal{S}_{static}$. A more thorough discussion on the patterns that can be generated using this approach can be found in the original paper.

This verification procedure, combined with the fact that the behavior deals with very limited robots (anonymous, homogeneous, memoryless, with limited range sensing, and without needing any communication, global knowledge, or seed robots) moving in space, sets the work by Coppola et al. (2019) apart from other works such as the ones of Klavins (2007), Yamins and Nagpal (2008), or Rubenstein et al. (2014). Furthermore, we remind the reader that here we are dealing with robots that should move in space such that the swarm arranges into a desired spatial configuration. This is different in nature from altering the internal states of the robots into a particular repeating pattern, such as in the work of Yamins and Nagpal (2008).

5.2 PageRank model

To define the PageRank framework for this task, we need to define graphs $G^a_{\mathcal{S}}$ and $G^p_{\mathcal{S}}$. This can be readily done based on the graphs $G^1_{\mathcal{S}}$, $G^2_{\mathcal{S}}$ and $G^3_{\mathcal{S}}$ introduced in Sect. 5.1.3. For $G^a_{\mathcal{S}}$, the graph holds the local transitions that a robot will experience based on actions it takes from a stochastic policy. This is exactly the same as $G^1_{\mathcal{S}}$, therefore:

$$G^a_{\mathcal{S}} = G^1_{\mathcal{S}}.$$

For $G^p_{\mathcal{S}}$, the graph holds all the local transitions that happen whenever the environment causes a state transition (i.e., other robots take an action). This can be the result of actions by neighbors, as described by $G^2_{\mathcal{S}}$, or actions of other robots which become neighbors, as described by $G^3_{\mathcal{S}}$. Therefore, it follows that:

$$G^p_{\mathcal{S}} = G^2_{\mathcal{S}} \cup G^3_{\mathcal{S}}.$$

The matrices $\mathbf{H}$, $\mathbf{E}$, and $\mathbf{D}$ are extracted from $G^a_{\mathcal{S}}$ and $G^p_{\mathcal{S}}$ using Eqs. 4, 5, and 6, respectively.

Concerning α, we follow the same definition as for the consensus task (see Eq. 9), whereby we define it based on the number of neighbors that a robot senses in a given state. This models how, with more neighbors, the robot is more likely to be subject to its environment. We once again have $0 \leq p_{action}(s) \leq 1$, where $p_{action}(s)$ is the cumulative probability of taking an action when in state s. If $p_{action}(s) < 1$, then there is also a probability that the robot will not take an action and that it will remain idle. In the example shown in Fig. 11a, for instance, we would have $p_{action}(s_i) = p_l + p_r \leq 1$. Here, p_l is the probability of taking an action to the left, and p_r is the probability of taking an action to the right, according to the policy that is being used.

[8] To this point, Condition 4 was made slightly more lenient than from what is expressed in (Coppola et al. 2019). The original version checks that a robot can travel to *all* open positions surrounding *all* of its neighbors. This is a strongly restrictive rule that heavily limits optimization. Therefore, we limited to checking that a robot could travel about all its neighbors, but not to *all* open grid positions around *all* its neighbors.

5.3 Optimization strategy

For this task, we wish to take care that the optimization procedure does not violate the conditions that tell us that the pattern will be formed, listed in Sect. 5.1.4. For this reason, we have divided the optimization process in two phases. In the first phase, we only perform the removal of state–action pairs from Π_0 while keeping the conditions from Sect. 5.1.4 as constraints. From this phase, we will extract a policy $\Pi_1 \subseteq \Pi_0$. After this is done, we remove the constraints and freely alter the probability of taking actions in Π_1, except that we always keep a nonzero probability for all state–action pairs. We would like to make it clear to the reader that Phase 1 is not required, and it is possible to directly go to Phase 2. (We provide an example of that in Sect. 5.7.) However, Phase 1 allows us to quickly reduce the solution space prior to Phase 2, while still ensuring that the pattern of interest can be formed. In both cases, we use the PageRank-based fitness function proposed in Eq. 7.

- **Phase 1: state–action pair removal from** Π_0 In Phase 1, state–action pairs are eliminated from the stochastic policy Π_0 with the goal of maximizing F. The input to this phase is the baseline policy that is extracted following the methods described in Sect. 5.1. The output of this first phase is a stochastic policy $\Pi_1 \subseteq \Pi_0$, whereby state–action pairs in Π_0 that do not help to achieve the pattern efficiently are automatically removed, while maximizing F as per Eq. 7. The optimization of Phase 1 is subject to the following two constraints:

  1. *The conditions from Sect. 5.1.4 must be respected* This constraint checks that the pattern will always eventually be formed from any initial configuration. The conditions can be checked based from the local graphs $G^1_{\mathcal{S}}$, $G^2_{\mathcal{S}}$, and $G^3_{\mathcal{S}}$, and therefore, the time required to check them scales with the local state-space size, and not the size of the swarm.
  2. *The final pattern must remain the unique emergent pattern* As state–action pairs are removed from Π_0, it may be that additional states behave like states in $\mathcal{S}_{static}$ and will not move. However, an important axiom needed to guarantee that P_{des} will always form is that, for a swarm of N robots, N instances of the local states in $\mathcal{S}_{static}$, with repetition, must uniquely rearrange into P_{des}. If this is not the case, another pattern could emerge where all robots are in a state $\mathcal{S}_{static}$ and do not move. It must therefore be verified that Π_0 is not affected in such a way that this can happen. The method to verify this is described at the end of this subsection.

- **Phase 2: probability optimization** In Phase 2, the probability of executing individual the state–action pairs in Π_1 is altered so as to maximize the fitness F. This phase parallels the final optimization step of Coppola and de Croon (2018), with the key difference being that we now evaluate the performance of swarm using the PageRank-based fitness function in Eq. 7, rather than by simulating the swarm. The output of this second phase is a stochastic policy Π_2.

Procedure to check the first constraint of Phase 1 Because we are already at the optimization stage, we consider a starting point whereby the original $\mathcal{S}_{static}$ already guarantees that P_{des} is unique. We then only need to check that adding new states to $\mathcal{S}_{static}$ does not affect this property, which we can do at the local level. Consider a state $s \in \mathcal{S}_{active}$ which has become a candidate to be moved to $\mathcal{S}_{static}$. For s, we locally check whether it could be fully surrounded by robots with a state within $\mathcal{S}_{static}$. If this is not possible, because s is such that at least one of its neighbors would be in an active state, then we can add s to $\mathcal{S}_{static}$. This is because we know that, if this state s were static, there would always be one active robot somewhere

Table 3 Genome size during Phase 1 for the four patterns tested

Pattern	Triangle (four robots)	Hexagon	Triangle (nine robots)	T (12 robots)
$\lvert\Pi_0\rvert$	187	486	531	525
$\lvert\Pi_1\rvert$	66	174	176	276

The genomes size is equal to the size of the original policy Π_0 extracted as described in Sect. 5.1.2. Following Phase 1, which could remove state–action pairs, the size of the policy could be reduced. These smaller policies can then be optimized in Phase 2

next to it anyway, so P_{des} still remains the only unique pattern that can be formed by static states. Furthermore, when this active neighbor moves, the local state of the robot will also change and it will also no longer be static. Note that, because of the importance of states in the $\mathcal{S}_{simplicial}$, these states are not allowed to transition into the set $\mathcal{S}_{static}$.

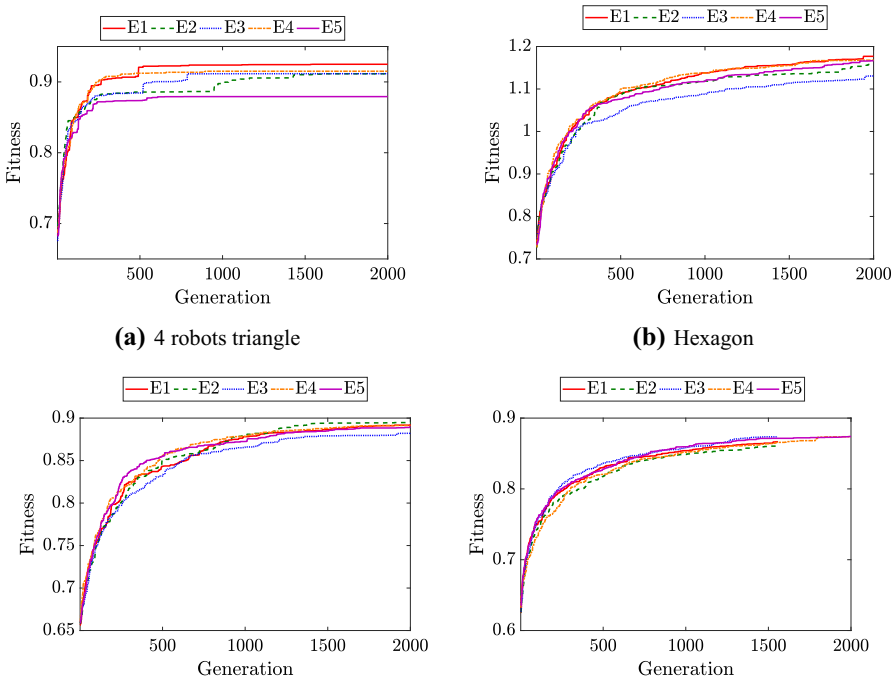
5.4 Phase 1: genetic algorithm setup and results

For Phase 1, we set up a GA with a population of ten binary genomes. Each gene in a genome represented a state–action pair in Π_0, with a 1 indicating that the state–action pair was kept and a 0 indicating that it was eliminated. Each generation consisted of: elite members (30%), new offspring (40%), and mutated members (30%). Offspring genomes were the result of an AND operation between two parent genomes. Offspring were only kept if they complied with the constraints, else the parents were forced to look for new mates. On each generation round, mutation was applied to a random portion of the population, for which the NOT operator was randomly applied to 10% of each selected member's genome (thus changing 1s to 0s and vice versa). Similarly as to the offspring, a mutation was kept if and only if it returned a genome for which the constraints were met, else it was discarded and a new mutation was attempted until a successful one was found. This way it was guaranteed that the population always consisted of genomes that respected the constraints. Each valid genome was evaluated assigned a fitness F to as per Eq. 7. The genomes of the initial population were generated by performing the mutation procedure on a fully positive genome (all 1s). The genomes size for the four patterns was equal to $\lvert\Pi_0\rvert$ for each pattern, as indicated in Table 3.

With this setup, five evolutionary runs were performed for the first four patterns of Fig. 9.[9] The results of the evolutionary runs are shown in Fig. 12. Within 2000 generations, the fitness of all patterns approached convergence for all evolutionary runs, characterized by a steep increase in the first generations.

The performance of all the evolved policies was evaluated in simulation. The simulator was a direct implementation of the swarm on a grid as described in Sect. 5.1.1. Each policy was evaluated 100 times with the swarm starting from random initial configurations, with the only requirement that they would begin in a connected topology. The global metric of performance was the amount of actions taken cumulatively by all robots before the desired final pattern emerged. This was compared to the original unoptimized policy Π_0 of each pattern. The results are shown as histograms in Fig. 13. It can be seen that all policies are able to achieve the pattern and significantly outperform the original unoptimized policy by approximately one order of magnitude, showing a successful application of PageRank as an

[9] As explained in Sect. 5.1.4, the line does not pass the verification procedure. It thus cannot be optimized in Phase 1. For this reason, it will be directly optimized using the GA of Phase 2, which does not directly enforce any constraints. The results for the line are shown separately in Sect. 5.6.



(a) 4 robots triangle

(b) Hexagon

(c) 9 robots triangle

(d) T with 12 robots

Fig. 12 Phase 1 evolutions for all patterns tested, showing the best fitness from each of the five evolutionary runs

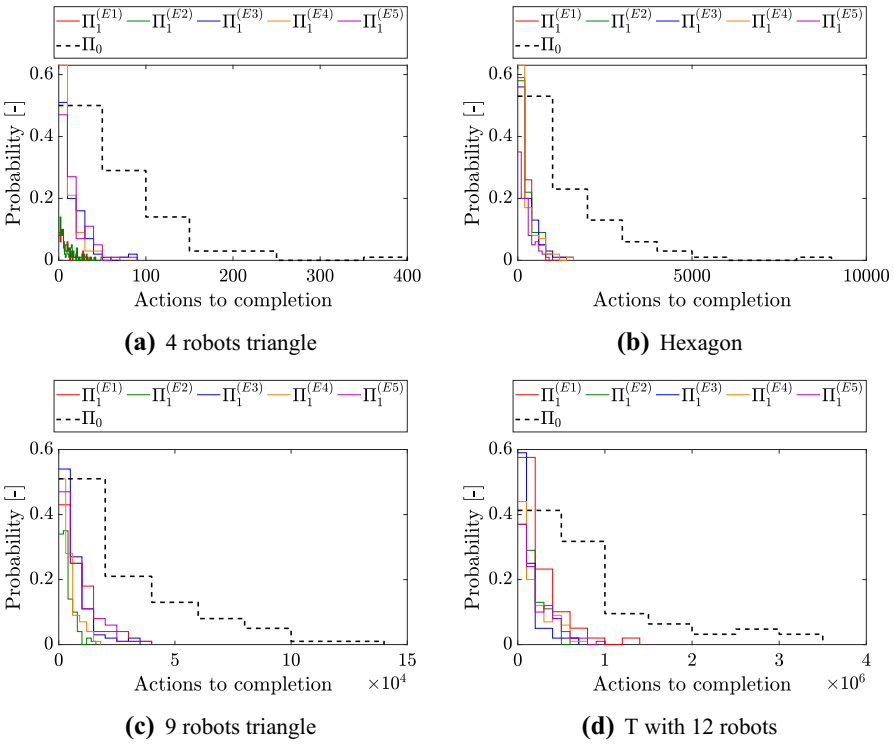
optimization parameter. The difference of the evolved performance from the original was confirmed by a bootstrap test with 10,000 repetitions, using the sample mean and the ratio of the variances as the reference statistics. All distributions were declared different over the 95% confidence interval. We now continue to Phase 2 in order to further improve the performance of the swarm.

5.5 Phase 2: genetic algorithm setup and results

Phase 1 can achieve a stochastic policy $\Pi_1 \subseteq \Pi_0$ which can be seen to provide a global performance that is one order of magnitude faster than Π_0. However, Π_1 still dictates that, when in a given state, a robot chooses from the available actions using a uniform distribution. This limitation is surpassed in Phase 2. The objective of Phase 2 is to optimize the probability of executing the state–action pairs in the policy, as was the case for the GA of the consensus task. The GA setup of Phase 2 is similar to the one of Phase 1, with two main differences:

1. The GA optimizes the probability of executing the state–action pairs from a stochastic policy Π_1, rather than determining which state–action pairs could be removed.
2. The constraints are no longer enforced (but, as none of the state–action pairs are removed, the provability is preserved).

For Phase 2, we set up a GA with a population of 20 scalar genomes. Each gene in a genome holds a value $0 < p \leq 1$, indicating the probability of executing the corresponding



(a) 4 robots triangle **(b)** Hexagon **(c)** 9 robots triangle **(d)** T with 12 robots

Fig. 13 Normalized histograms showing the cumulative number of actions required before the global goal is achieved for the evolved policies $\Pi_1 \subseteq \Pi_0$, extracted from the five evolutionary runs of Phase 1 (colored lines), compared to a baseline unoptimized performance of Π_0 for each pattern (black line) (Color figure online)

state–action in Π_1. Naturally, for each state, the cumulative probability of taking one of the actions in Π_1, denoted $p_{action}(s)$, cannot be larger than 1. If this happens, then the probabilities of the relevant state–action pairs in the genome are appropriately normalized. Each new generation was produced by elite members (30%), offspring (40%), and mutated members (30%). Offspring were generated from averaging two parents' genomes. Mutation was applied to random genomes, for which 10% of their genes were replaced by random values of $0 < p \leq 1$ from a uniform distribution. The members of the initial population were produced by applying the mutation procedure above to a genome with uniform probability distributions, as extracted from Phase 1. The genome size is detailed in Table 3.

Five evolutionary runs were conducted for each pattern for 1000 generations. The policy to be optimized was the best performing policy from Phase 1, for each pattern, based on the results of Phase 1 shown in Fig. 13. The results of the evolution of Phase 2 are shown in Fig. 14. The global performance of the final policies Π_2 was evaluated 100 times as in Phase 1. Once again, we measure the global performance by the cumulative number of actions that the robots in the swarm take before the desired pattern emerges. The results are shown in Fig. 15, where they are compared to the best policy obtained at the end Phase 1 for each pattern. For the triangle with four robots and the hexagon, we also compare these results to the performance of the optimized behavior from Coppola and de Croon (2018), which used simulation in order to directly optimize for the cumulative number of actions taken by the swarm. Sample trajectories of simulations are shown in Fig. 16.

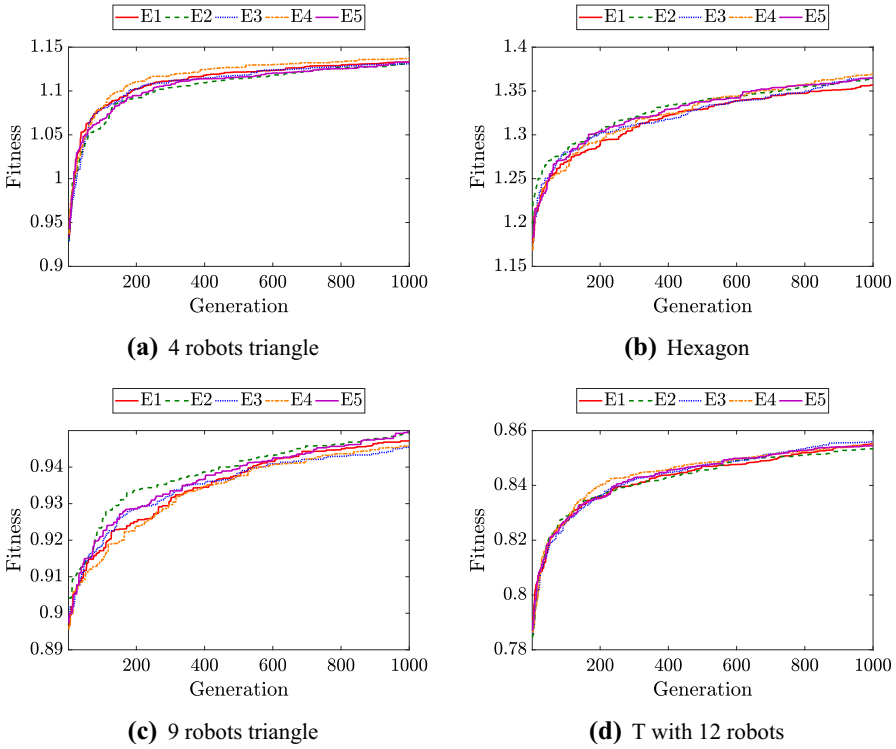
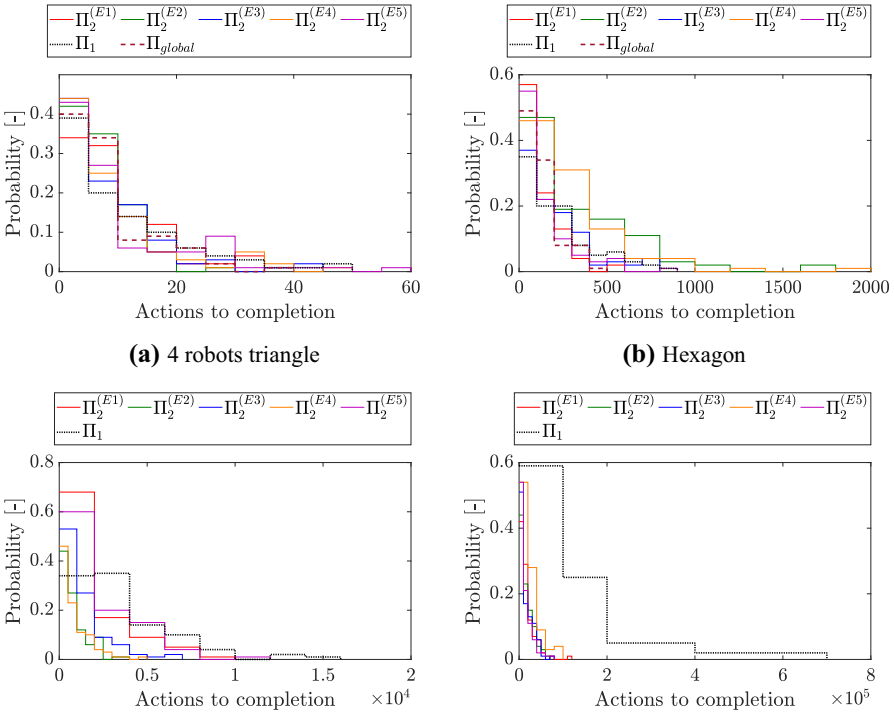


Fig. 14 Phase 2 evolutions for all patterns tested, showing the best fitness from each of the five evolutionary runs

The improvements achieved for the triangle with four robots and the hexagon appear marginal. For two policies evolved for triangle with four robots, a bootstrap test (based on the difference in sample means) could not reject the hypothesis that the distribution of the final performance is equal to the performance at the end of Phase 1. This was the case for also one of the policies evolved for the hexagon. However, this is a justified result, because for both the triangle with four robots and the hexagon, the performance after Phase 1 already approached the performance of a policy which was optimized with the fitness being evaluated through simulations, which we take to be a near-limit performance. The evolutions of Phase 2 achieved policies that performed equivalently and, in some cases, seemingly even slightly better than the ones that were achieved in Coppola and de Croon (2018). In Fig. 15, it can be seen that a higher fraction of runs take fewer than ten actions and fewer than 100 actions for the triangle with four robots and the hexagon, respectively. These results show the power of PageRank's framework in being able to describe the swarm up to the point that the achieved performance matches the performance of results that are achieved with a global evaluation procedure.

For the triangle with nine robots and the T with 12 robots, Phase 2 instead shows significant improvements over Phase 1. These are most noticeable for the T with 12 robots, which reaches an improvement in performance of one order of magnitude compared to the policies from Phase 1. Overall, this means that the policies following Phase 2 outperform the unoptimized policy by two orders of magnitude. We remind the reader that both of these patterns could



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

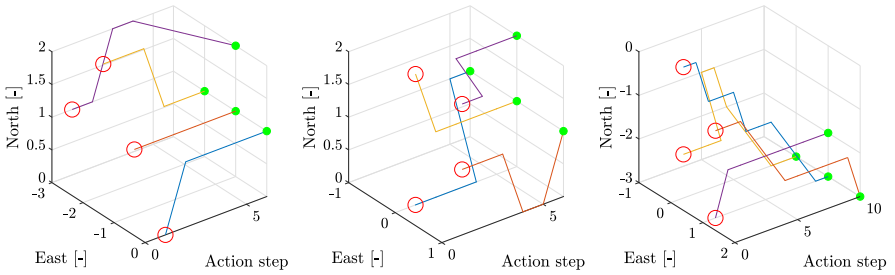
(c) 9 robots triangle **(d)** T with 12 robots

Fig. 15 Normalized histograms showing the cumulative number of actions required before the global goal is achieved for the evolved policies from Phase 2, in comparison with the best performing policy of Π_1, and the globally optimized performance from Coppola and de Croon (2018). The latter is available only for the first two cases, the triangle with four robots and the hexagon, as extracting it for larger patterns leads to scalability issues)

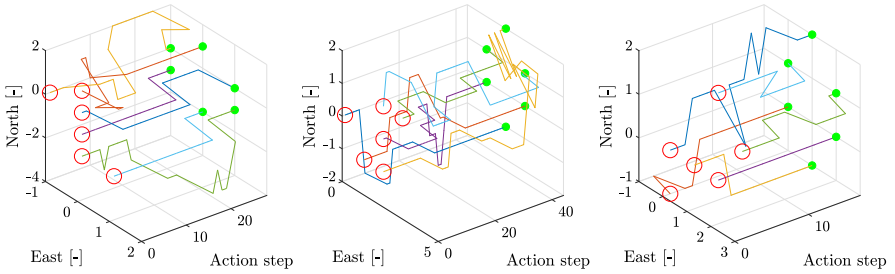
never have been tackled if, instead of using PageRank, we had used simulation in order to assess the performance of the swarm. Simulations would have been infeasible in light of scalability issues, as was indeed the case in Coppola and de Croon (2018). Instead, these results have been achieved with the use of an entirely local evaluation.

5.6 Analysis

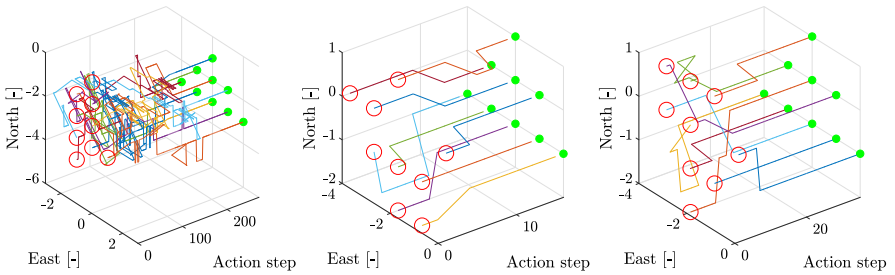
The results of Phase 1 and Phase 2 both show that the evolutions successfully lead to policies with high performance boosts, which even match the performance of policies that were optimized using a fitness function based on a global simulation of the swarm. In this section, we shall now aim to gain more insight into how the PageRank-based fitness function was correlated with the global performance of the swarm. As we are now dealing with a large policy with a large number of parameters (see Table 3), the entire fitness landscape cannot be studied. Instead, we evaluated the global performance at various stages of the Phase 1 and Phase 2 evolutions to show how the performance improvements are correlated with the PageRank-based fitness.



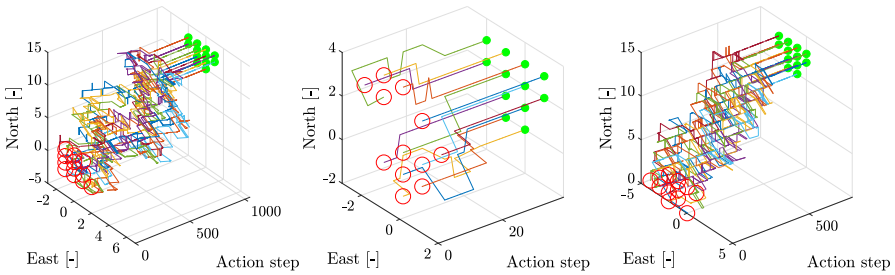
(a) 4 robots triangle



(b) Hexagon

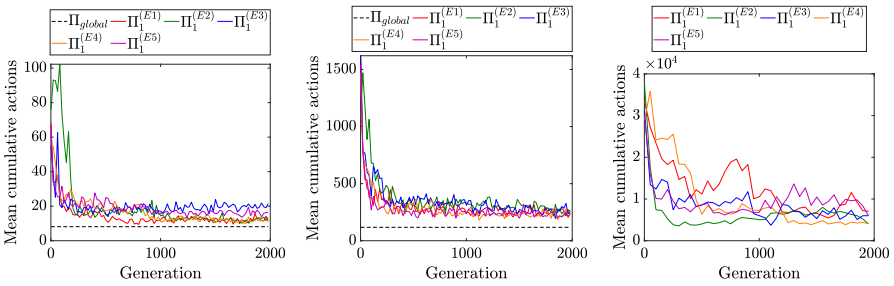


(c) 9 robots triangle



(d) T with 12 robots

Fig. 16 Exemplary simulations showing the evolution of the swarm with policy Π_2 for each pattern studied, given random initial conditions. The red dots indicate starting positions, and the green dots indicate final positions (Color figure online)



(a) 4 robots triangle **(b)** Hexagon **(c)** 9 robots triangle

Fig. 17 Evaluation of the average performance improvement during Phase 1 for all evolutionary runs. The performance is measured as the average number of actions required cumulatively by the swarm in order to achieve the pattern. This is based on 100 simulations for the highest fitness at each generation, sampled every 20 generations for the triangle with four robots and the hexagon, and every 50 generations for the triangle with nine robots. The dotted line shows the mean performance of the policy achieved by a global optimization in Coppola and de Croon (2018), only available for the first two patterns

For Phase 1, the results of this evaluation are shown in Fig. 17. It can be seen that the performance of the swarm improves quickly in the beginning. This matches the fast rise in fitness observed in Fig. 12. Moreover, for the triangle with four robots and the hexagon, we see that the performance even begins to approach the one of the final global optimization from Coppola and de Croon (2018). Overall, Phase 1 shows a successful correlation between the PageRank-based fitness function and the global performance of the swarm.[10] For Phase 2, the results of the same analysis for the triangle with four robots, the hexagon, and the triangle with nine robots are shown in Fig. 18. Once again, it is possible to observe a general overall improvement in the performance. In all cases, over the first generations, the performance of the swarm improves.

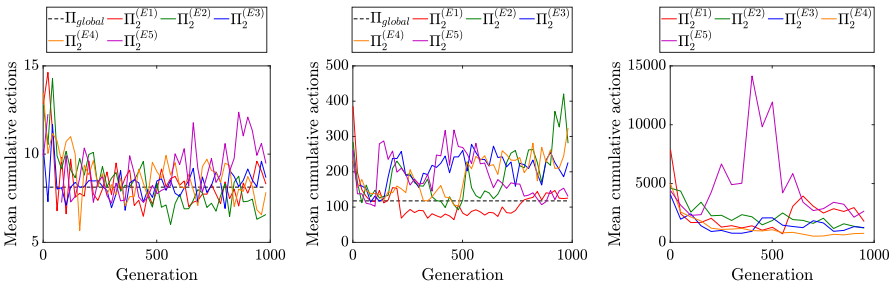
It is in Phase 2 that, however, we also notice certain limitations. It appears that, beyond a certain point, some evolutionary runs begin performing worse as the evolution continues, which is not reflected in the fitness function. The effect is most clearly noticeable on:

– the hexagon, starting from around the 100th generation, for all runs.
– the triangle with nine robots, where one of the evolutionary runs features a drastic spike in performance at approximately the 400th generation (after which it seems to recover).

These effects are likely the result of reaching a limitation of PageRank's microscopic model once the constraints from Phase 1 are removed. Eventually, there comes a point where the evolution over-fits to the microscopic model at the expense of the global performance. There are also a number of other effects at play:

1. The change in fitness is low compared to Phase 1 evolutions; hence, the effects of noise are more clearly noticeable.
2. The impact of changing probabilities in a stochastic policy is less direct than completely removing a state–action pair from the policy. Therefore, the changes have less impact on the results.

[10] Unfortunately, we cannot show this analysis for the T with 12 robots. This is because evaluating early generations in simulation is too computationally expensive. This serves as another reminder of how infeasible it would have been to try and optimize the behavior of the swarm using simulations to assess the fitness. Only the results before and after the evolutions have been evaluated, which can be appreciated in Figs. 13d and 15d.



(a) 4 robots triangle **(b)** Hexagon **(c)** 9 robots triangle

Fig. 18 Evaluation of performance improvement during Phase 2 for all evolutionary runs. The performance is measured as the average number of actions required cumulatively by the swarm in order to achieve the pattern. This is based on 100 simulations for the highest fitness at each generation, sampled every 20 generations for the triangle with four robots and the hexagon, and every 50 generations for the triangle with nine robots. The dotted line shows the mean performance of the policy achieved by a global optimization in Coppola and de Croon (2018), only available for the first two patterns

5.7 Further investigations

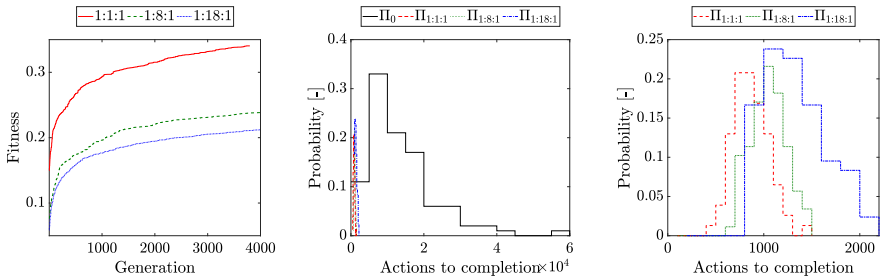
We conclude this section with a further investigation, whereby we explore whether placing more importance on certain desired states can be beneficial or destructive to the pattern formation task. This is relevant to patterns that feature a repeating desired state.

Consider, for example, the line along northeast as shown in Fig. 9. The line only requires three states in the set $\mathcal{S}_{des}$ (bottom edge, middle piece, and top edge), yet can be generated with any number of robots by nature of a repeating middle state. We thus investigated whether, for long lines, placing more importance on the repeating middle state would improve the performance of the optimized policy. This was done with an altered version of the fitness function which calculated a weighted average of the desired states, rather than a normal average, as in Eq. 10.

$$F_w = \frac{\sum_{s \in \mathcal{S}_{des}} w(s) * R(s)/|\mathcal{S}_{des}|}{\sum_{s \in \mathcal{S}} R(s)/|\mathcal{S}|}, \tag{10}$$

where $w(s)$ indicates the importance of a state $s \in \mathcal{S}_{des}$. We performed three different optimization runs with weight ratios 1:1:1, 1:8:1, and 1:18:1, relating to the weight of the bottom edge, middle piece, and top edge, respectively. For these optimizations, we directly used the optimization setup of Phase 2, with the fitness function of Eq. 10. The results of the evolution are shown in Fig. 19a. When the performance of the obtained policies was evaluated for a swarm of 20 robots, it was found that the policy that was optimized with a weight ratio of 1:1:1 featured the best performance, followed by 1:8:1, and finally by 1:18:1. This is shown in Fig. 19b, c.[11] These results lead to the insight that, at least for this type of task, it appears equally important for robots to reach all desired states simultaneously, and preference toward one may be detrimental.

[11] Note: creating a line along northeast is subject to both spurious patterns and livelocks. These, which happened on occasion, have been removed from the results shown in these figures, as they are outside the scope of this discussion.



(a) Results of evolutions of ≈4000 generations

(b) Performance compared to original unoptimized performance when generating a line with 20 robots

(c) Relative performance between three evolved policies

Fig. 19 Study on the impact of changing the relative importance of states in $\mathcal{S}_{des}$. Even when a state repeats, the performance remains more optimal when all desired states are weighted equally in the fitness function

6 Sample task 3: aggregation

In this task, robots move within a closed arena with the goal of forming aggregates. Each robot can sense the location of neighbors within a range. This task is similar in nature to that of Correll and Martinoli (2011). Using PageRank, we optimize a stochastic policy that tells a robot whether it should perform a random walk or remain idle based on the number of neighbors that it can sense. The final policy is optimized without taking into account the lower-level implementation of the random walk, the number of robots in the swarm, the size of the environment, or the initial condition. Applying the framework to optimize a higher-level policy such as this one provides further insight into the applicability and flexibility of the approach.

6.1 Task description

Consider N robots in a closed arena. The robots operate in continuous space (rather than a grid, as in previous tasks), and in continuous time, using local clocks. Each robot can sense the number of neighbors, m, within a limited range $\rho_{\max}$. We assume that each robot is capable of sensing a maximum of M neighbors, after which the sensor saturates. Therefore, the local state space $\mathcal{S}$ for this task is given by all possible numbers of neighbors that can be sensed by the robot, namely $\mathcal{S} = \{0, 1, \ldots, M\}$. The sensor updates its reading with a time interval t_c.

Based on the number of neighbors sensed, a robot can choose to engage in one of the following two sub-behaviors:

1. Perform/resume a random walk behavior. We denote this action as a_{walk}.
2. Stop and remain idle. We denote this action as a_{idle}.

These form the action space $\mathcal{A} = \{a_{walk}, a_{idle}\}$. The policy Π, to be optimized using the PageRank algorithm, is the stochastic policy mapping the state space $\mathcal{S}$ to the actions space $\mathcal{A}$. The set $\mathcal{S}_{des} = \{m_{\min}, m_{\min} + 1, \ldots, M\}$ is the set of desired states, where $m_{\min}$ is the minimum amount of neighbors that we wish for a robot to observe.

6.2 PageRank model

To set up the PageRank framework, we begin by defining the active graph $G^a_{\mathcal{S}}$, which describes the local state changes as a result of the actions of a robot.

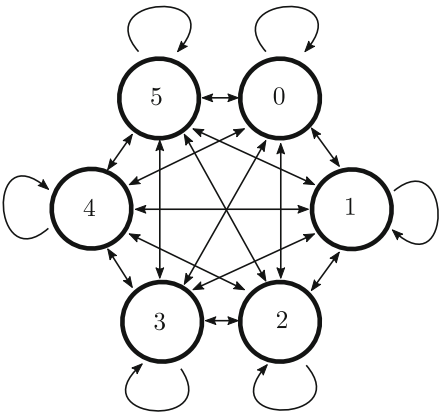
The graph $G^a_{\mathcal{S}}$ is defined as a fully connected graph between all states, as exemplified in Fig. 20a for the case where $M = 5$. If the robot remains idle, then it will always keep its number of neighbors (remember that all changes in the environment, including neighbors leaving, are described in the passive graph, $G^p_{\mathcal{S}}$). We thus set the weights for all transitions to the same state to be equal to $\Pi(s_i, a_{idle})$. All other edges are given weights according to the following:

$$w_{i,m} = \Pi(s_i, a_{walk}) \cdot p(m), \tag{11}$$

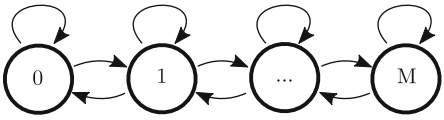
where $p(m)$ is the probability of transitioning to a state with m neighbors. In our work, we set: $p(m) = 1/(m+1)^2$, albeit we have found that the same final policy is evolved also for other definitions, such as $p(m) = 1/(m+1)$. Therefore, we can see that it is sufficient to coarsely model the trend that the probability of finding larger groups decreases with the size of the group. The swarm designer does not need to concern oneself with the exact specification of how likely it is to encounter larger group sizes, which eases the design task.

Graph $G^p_{\mathcal{S}}$ holds the transitions that a robot can experience regardless of its own choice to be idle or not. In this case, the robot can be in any state $s \in \mathcal{S}$, at which point it could lose a neighbor or gain a neighbor at any point. The graph $G^p_{\mathcal{S}}$ takes the form shown in Fig. 20b.

As done for the previous tasks, α is set as a function of the total probability of selecting a new action and the number of neighbors, as in Eq. 9. We also note here, however, that our evolutions returned the same final behavior also for simpler static definitions of α, such as $\alpha = 0.5$, which would simply model that at all times the robot is equally subject to its decisions as it is to its environment. This is likely because the behavior that is being optimized is too high level for such details to make a real difference.

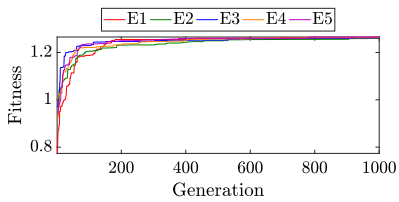


(a) Depiction of $G^a_{\mathcal{S}}$ for the sample case where $M = 5$.

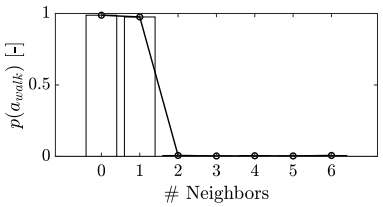


(b) Depiction of $G^p_{\mathcal{S}}$ for an arbitrary value of M. All transitions from any node are equally probable. To model random concurrency, it is assumed that two robots will never leave (or join) a neighborhood at exactly the same time.

Fig. 20 Depictions of $G^a_{\mathcal{S}}$ and $G^p_{\mathcal{S}}$ models for the aggregation task



(a) Fitness over five evolutionary runs, showing the best fitness for each run.



(b) A final evolved policy, showing the probability of moving with respect to the number of neighbors that are sensed. The robots almost always move if the number of neighbors found is less than the desired amount, else they always remain idle.

Fig. 21 Details of the evolution of the policy for the aggregation task, showing the increase in fitness a final evolved policy

6.3 Genetic algorithm setup

We used the same GA setup as used for the consensus task in order to tune the probabilities of the policy Π for this task. We considered the case where $m_{\min} = 2$, meaning that the robots have the goal of forming aggregates of three or more. Unlike other cases, we did not set the policy values for the states $\mathcal{S}_{des}$ to zero. Note that this is not necessary and actually makes the problem harder to solve, since more freedom is given to the optimizer with more parameters to be optimized. Setting the policy values for the states $\mathcal{S}_{des}$ would have automatically forced a relative basic solution (i.e., "move when not in a group") upon the behavior. Thus, we instead took this as an opportunity to see whether PageRank would be able to tune its weights to also inform the robots to remain in the states of $\mathcal{S}_{des}$, without us providing this knowledge a priori.

The results of five evolutionary runs, showing the best genome for each generation, are shown in Fig. 21a. The evolved behavior was the same in all cases. The behavior we extracted from one evolutionary run is shown in Fig. 21b. Here, we can clearly see that the final evolved behavior is simple, yet very sensible. The extracted policy approaches a deterministic policy whereby if the number of neighbors is less than $m_{\min} = 2$, then the robot always chooses to move and explore the environment in the hope of finding more neighbors. If the number of neighbors is sufficient, then the robot remains idle.

6.4 Simulation environment and evaluation results

The performance of the evolved behavior was tested using a simulation environment called *Swarmulator*. Swarmulator is a C++-based simulator designed to efficiently simulate and prototype the behavior of swarms in continuous time and space. Each robot (together with its dynamics) is simulated and controlled by an independent detached thread, which limits simulation artifacts which may otherwise stem, for instance, from the robots being simulated always in the same order. In our simulations, the robots had the same dynamics as accelerated particles, allowing them to move omnidirectionally. The details of the simulated behavior were the following:

– When initiating a random walk, a robot could randomly select a direction θ from a uniform distribution $[0,\ 2\pi]$ and pursue it with a reference speed v. The decision to

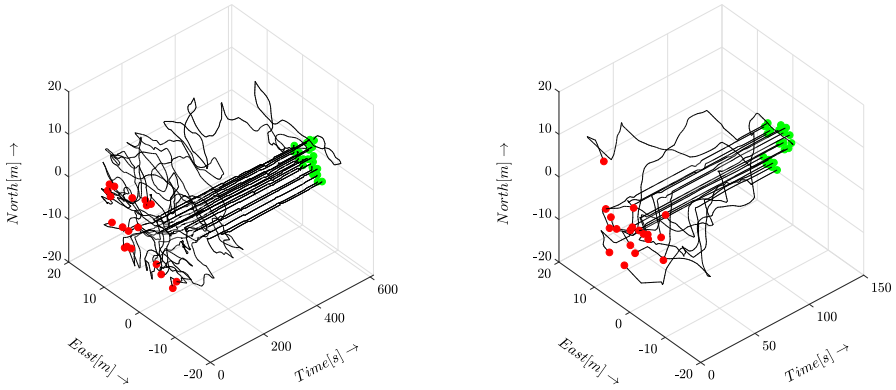


Fig. 22 Depiction of two simulations of the aggregation task using the evolved policy together with the simulation environment described in Sect. 6.4. On the left figure, the robots assemble into one cluster. On the right figure, the robots assemble into multiple clusters. Multiple clusters may happen due to the low sensing range of the robots in comparison with the size of the arena (Correll and Martinoli 2011)

continue the random walk or stop, commanded by the high-level policy, was reassessed with an interval t_c. If the robot chose to remain in a random walk, then it altered its current direction θ by an angle θ_d. The angle θ_d was extracted from a normal distribution with zero mean and a standard deviation σ_d. In our implementation, we set $v = 0.5\,\text{m/s}$, $t_c = 2\,\text{s}$, and $\sigma_\theta = 0.3\,\text{rad}$. The robots could sense their neighbors omnidirectionally up to a distance $\rho_{\max} = 2\,\text{m}$.

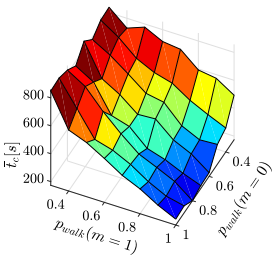
– If a robot was at a distance d_w from a wall, it would set a velocity command away from the wall for a time t_w. In our implementation, we set $d_w = \rho_{\max} = 2\,\text{m}$ and $t_w = 2t_c$. Setting $d_w = \rho_{\max}$ simulates the fact that the same sensor is used to sense both neighbors and the walls.
– At all times, in direct addition to the velocity commands from the random walk behavior and wall avoidance, the robots were governed by attraction and repulsion forces. This handled collision avoidance. In our implementation, the attraction and repulsion behavior was tuned such that the equilibrium distance between robots was 1.5 m.

The system above was tested 50 times for a swarm of 20 robots in a $30\,\text{m} \times 30\,\text{m}$ square arena. In all cases, the swarm successfully forms aggregates of three or more robots within the arena. Two sample runs are shown in Fig. 22.

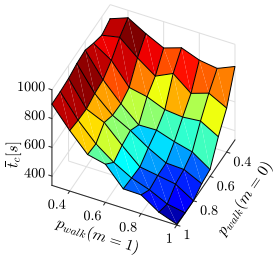
6.5 Analysis

As for the variant of the consensus task from Sects. 4.4 and 4.5, we took advantage of the small size of this policy in order to map and investigate the fitness landscape. The investigation was focused on testing the impact of the probability of moving within the arena when the robots did not have any neighbors and when the robot had one neighbor. Since the robots had the goal of being with two or more neighbors, these were the two main parameters of interest.

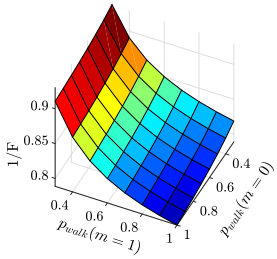
The simulation environment from Sect. 6.4 was used to evaluate the performance of the swarm against different policies, in order to construct the fitness landscape with respect to our global parameter of interest. The results were obtained by averaging 50 simulations for each setting, and repeated for swarms of 20 robots and 50 robots in a $30\,\text{m} \times 30\,\text{m}$ arena.



(a) Global performance of the solution space for a swarm of 20 robots in a 30 m × 30 m arena. $\overline{t_c}$ is the mean time taken before the task is completed.



(b) Global performance of the solution space for a swarm of 50 robots in a 30 m × 30 m arena. $\overline{t_c}$ is the mean time taken before the task is completed.



(c) PageRank-based fitness landscape, where $1/\mathrm{F}$ is the inverse of the fitness evaluated through Eq. 7.

Fig. 23 Global mapping of the solution space for the aggregation policy for different probabilities of motion with zero and one neighbors. $p_{walk}(m)$ is the probability of pursuing a random walk when surrounded by m neighbors

The maximum simulated time allowed for the task was set to 1000 s. The results are shown in Fig. 23a, c, respectively.

This fitness landscape, which is the global metric that we wanted to optimize for in reality, is compared to the fitness landscape shaped by the PageRank-based fitness function with which the behavior was actually evolved. The fitness landscape, shown in Fig. 23c, has a clear correlation with the global performance landscapes shown in the remainder of Fig. 23. The result of this analysis shows that the behavior that was evolved approaches an optimum that is also found at the global level.

7 Discussion

In Sects. 5, 4, and 6, the PageRank framework has been applied to optimize the stochastic policy of robots in a swarm for three separate tasks. With the knowledge gained, this section summarizes the advantages of the proposed approach (Sect. 7.1), its current limitations, and possible solutions (Sect. 7.2) and proposes research directions that could lead to interesting extensions (Sect. 7.3).

7.1 Advantages and properties of the proposed approach

The primary motivation for the development of the approach proposed in this paper was to find a method for which the time required to evaluate a behavior does not scale with the size of the swarm. Taking as example the tasks tested in this paper, if the fitness of the controller were to have been tested in simulation, then the time to be evaluated (wall-clock time) would have been dependent on a number of parameters, such as:

- The number of robots in the swarm, which affects both the processing time required to simulate it and the time before which the global goal is achieved.
- The size of the environment, which would alter the rate at which events occur. This would have been relevant for a task such as the aggregation task.

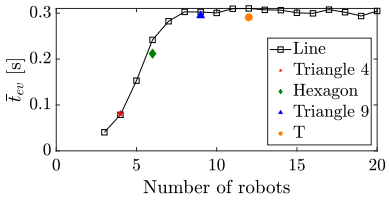
Furthermore, the final evolved solution would have been fitted to the following:

- The initial conditions from which the swarm is simulated. To avoid over fitting to a particular initial condition, each controller would have had to be tested from several random initial conditions.
- The number of robots used in the simulations. To avoid over fitting to swarms of a specific size, each controller would have had to be tested on swarms of different sizes.
- The global environment. If evolving within a specific environment, the final solution may implicitly be fitted to it.

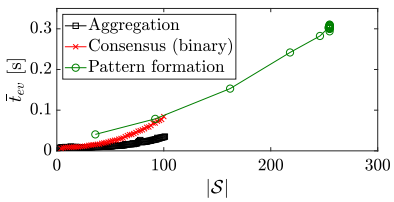
Certain methods may help to mitigate the issues above when simulation is used, such as evolving for fewer robots and then testing on larger swarms (Trianni et al. 2006), or capping the simulation time to a maximum value (Scheper and de Croon 2016). Nevertheless, such solutions present new challenges of their own. The first, for instance, does not guarantee that the controller will be scalable. The latter, on the other hand, may limit the ability of the evolutionary algorithm to properly search the solution space. If the time limit is too low, then few genomes, if any, will succeed in the goal. Moreover, this means that the fitness is no longer assessed by whether, and how efficiently, the goal is achieved, but must be assessed based on how much of the goal has been achieved. This may require a different fitness function, a task which can likely cause additional difficulties and may even shape the final outcome (Nelson et al. 2009).

Using PageRank, the evaluation of the fitness function is based only on parameters extracted from a microscopic model. This has the following beneficial effects, which can be found back in the results of this paper:

- **The wall-clock time does not scale with the size of the swarm** This can be appreciated directly from the results in this manuscript, such as for the pattern formation task, where we have managed to optimize the policy for patterns that we could not tackle using simulation to evaluate the fitness (Coppola and de Croon 2018). We have plotted the wall-clock time required to evaluate the PageRank-based fitness for the pattern formation task in Fig. 24a. Each data point is averaged over 100 evaluations of PageRank for random instances of a stochastic policy. It can be seen that, once the maximum neighborhood size of 8 is fulfilled, the wall-clock time stops scaling and it remains constant even if the number of robots in the swarm increases. Instead of scaling with global parameters, the wall-clock time scales with the number of local states $\mathcal{S}$ and the possible transitions between them, as modeled by $G^a_{\mathcal{S}}$ and $G^p_{\mathcal{S}}$. To test the practical impact of this, we evaluated the wall-clock time of the PageRank-based fitness function when varying the size of $\mathcal{S}$ (which also bring about more state transitions) for the separate tasks tested in this paper, averaged over 100 random policy iterations. The results are shown in Fig. 24b. It can be seen that the wall-clock time grows with the size of $\mathcal{S}$. In comparison, the fitness wall-clock time for simulations directly grows with the size of the swarm. This is shown for the pattern formation task from Fig. 25. Note that this is despite using the simple grid simulation environment as was used for the simulations in Sects. 5.4, 5.5, and 5.6. The performance of higher fidelity simulators would be worse.
- **The evolved policy scales well with the number of robots in the swarm** The policy is evaluated from a graph for which the number of robots in the swarm is not modeled, only the probability that certain events in the neighborhood of a specific robot may take place. For example, for the consensus task, the same exact policy could be applied to swarms of different sizes (we tested with swarms of ten and 20 robots), from random initial configurations, and consistently provided efficient results orders of magnitude better than the baseline case.



(a) Change in wall-clock time with the number of robots for the pattern formation task. Once the swarm reaches 9 or more robots (such that the local neighborhood of a robot could be entirely filled with 8 other robots) the wall-clock time does not scale.



(b) Change in wall-clock time as the size of the local state space increases.

Fig. 24 Trends in wall-clock time of PageRank-based fitness evaluation, denoted $t_e v$. When using the PageRank centrality measure, the wall-clock time scales with the size of the local state space. $\bar{t}_{ev}$ denotes the average wall-clock time over 100 random policy evaluations. These have been computed using a MATLAB script on a Dell latitude 7490 laptop with Intel®Core™i7-6600U CPU @ 2.60GHz × 4, Intel®HD Graphics 520 (Skylake GT2), 8GB RAM, equipped with SSD, and running Ubuntu 16.04

- **The evolved policy is not tuned to any particular initial condition of the swarm** The proposed PageRank-based fitness function (Eq. 7) evaluates the likelihood of achieving certain local states. However, as PageRank is evaluated for the entire graph simultaneously using the iterative procedure described in Sect. 3, the score does not assume any initial condition by the robot, and, by extension, the swarm. This makes it robust to the initial conditions. In all results shown in this paper, there was no initial condition for which the evolved solution made it impossible to achieve the final global goal or for which the performance did not improve.
- **The evolved policy is not tuned to any particular lower-level controller** In all instances, a policy was evolved without any specification of lower-level parameters. This can be best appreciated for the aggregation task. Here, we evolved a high-level policy without specifying the type of random walk that was going to be implemented, but only specifying the relative probability of encountering neighbors when a robot is moving. Had such a behavior been tuned in simulation, then it would have inherently adapted to the particular implementation of search behavior (e.g., wall-following, Lévy Flight, etc.), whereas this is not the case for our results, providing a more general controller. The relative probability of encountering neighbors subsumes a larger set of specific lower-level behaviors. Similarly, for the consensus task and the pattern formation task, the behavior is evolved without any particular specification of communication time, decision time, or motion controller. A simulation-based optimization might have adapted the probabilities to implicitly account for these specifications. Instead, the only concern for the PageRank-based optimization is with which probability it should transition between states.
- **The evolved policy is not tuned to any particular global environment** PageRank's framework only models the possible transitions that may happen in the environment, but not the entire environment itself. Therefore, as long as the possible local transitions that may take place in a global environment are sufficiently well represented, then the evolved policy remains robust to different environments. In some instances, the global environment may also be excluded altogether. This was the case for the consensus task and the pattern formation task.

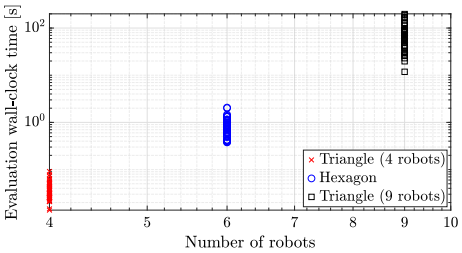


Fig. 25 Trends in wall-clock time to evaluate the performance of a policy in a simulation environment for the pattern formation task, evaluated for 50 randomly generated policies for which the wall-clock time was averaged over ten iterations. Note that the plot features a log scale on both axes. The simulator used was a simple grid simulation implemented in MATLAB. The machine details are the same as used for the results in Fig. 24

7.2 Current limitations and potential solutions

We have identified a set of limitations that pertain to the current implementation of the proposed approach. These are discussed below. When relevant, possible solutions or extensions are proposed.

- **Discrete state space and discrete action space** The current version of the approach relies on a discrete set of local states and a discrete set of actions. This may be limiting for applications with continuous states. The limitation comes from the fact that we are using a tabular policy which maps discrete states to discrete actions, which then allows us to construct a graph with discrete nodes and edges. Although this is a limitation of the current form, future work can explore the possibilities of extending the approach to deal with continuous state spaces. A research direction that would be interesting to follow is the possibility to use basis functions to interpolate over continuous state spaces. This type of solution has been proposed in the past for other approaches that also natively rely on a tabular policy, such as Q-learning, in order to accommodate continuous state spaces and/or action spaces (Sutton et al. 1998; Busoniu et al. 2010).
- **"Missing out" on the global optimum** Although the benefits of swarming are often credited as flexibility, robustness, and scalability (Hamann 2018), some applications may benefit from using policies which are tuned to a specific settings. For instance, if it is known a priori that a swarm of robots will always exclusively operate within a specific environment and be composed of a fixed number of robots, then it may be desired to find the global optimum for this particular setup. As an example, consider an aggregation task with a swarm of three robots. An optimal policy might feature that it is beneficial for two robots to wait for the third to come into their neighborhood. The number of robots would thus be implicitly encoded into the final policy. Instead, the locally evolved policy would not consider this option, as the number of robots would not be known nor modeled explicitly throughout the evolution. Although this means that the solution is likely more scalable, flexible, or robust, it does mean that it is less globally optimal for the specific case. Moreover, as the approach is local in nature, it is not possible to directly evaluate global swarm parameters such as the ideal number of robots for a given task.
- **The need to define desired local states** A complexity and potential limitation of using our approach is the need to define the set $\mathcal{S}_{des}$. This is the set of locally desired states that form the global goal. For the three tasks shown in this paper, $\mathcal{S}_{des}$ was extracted with ease using intuition. Nevertheless, as task complexity grows, $\mathcal{S}_{des}$ may become more difficult

to define. A potential solution to this problem could be to use an automatic method which is capable of inferring the local desired states by observing a global behavior. This is discussed further in Sect. 7.3.

– **Assumption of successful state transitions** The microscopic model used to evaluate PageRank assumes that state transitions are successful in executing the policy. In reality, this may not always be the case in light of imperfect actuation or noisy sensors, which may cause unexpected state transitions that are not modeled. This limitation can be addressed at the robot level via proper filtering of sensory data as well as accurate lower-level control. From the policy optimization side, it would be interesting to see whether this issue can be tackled via an online refinement of the model, allowing the policy to adapt itself to the imperfections of the real world during deployment. We continue this discussion in Sect. 7.3.

7.3 Further potential extensions

– **Extension to online learning** The current implementation uses an offline learning process, whereby the graphs G_S^a and G_S^p are devised based on a model of the robot and the task. There is also the possibility to extend the approach to online learning. The graphs G_S^a and G_S^p are based on the expected local states and transitions that the robot can experience. All these local states and all these transitions can be measured by the local sensors present on the robot. Therefore, this means that it is also possible for a robot to construct, expand, and/or tune the graphs G_S^a and G_S^p online directly based on its experiences during operation. In turn, it could then re-optimize its controller to suit these new graphs. For example, this could be done in order to automatically tune to the failure probability of certain actions using an online refinement of the transition rates. It can also be used to tune against "noisy" actuators which fail (with a given measurable probability) to follow through on an action, and either do not complete the action or create a different state transition than planned. Alternatively, it could adapt to systematic events in its environment, placing a higher probability on those transitions in the graph G_S^p. The parameter vector $\boldsymbol{\alpha}_\mathbf{v}$ could also be estimated or refined online. Thanks to the generally low computational efforts required to run the optimization (notice the wall-clock times in Fig. 24), the optimization could also be performed on-board of a robot without requiring excessive computational power.
– **Alternative fitness functions** The fitness function proposed in this work (Eq. 7) focused on maximizing the probability that a robot in the swarm would be in a desired state. The hypothesis that this would lead to global level improvements was found to be successful for each of the tested scenarios. Further investigations led us to the discovery that there may also be correlations to other parameters stemming from PageRank. For instance, using the results of the analysis from the pattern formation task from Sect. 5.6, we have found that the variance of the centrality also appears to show a correlation with the global performance of the swarm. In future work, it would be interesting to see whether, and how, alternative fitness functions could be constructed in order to provide even larger global level benefits.
– **Extension to heterogeneous swarms** This paper has tested the algorithm on homogeneous swarms of robots with one global goal. However, we wish to note that swarm homogeneity is not a direct limitation of the proposed algorithm. In fact, the "passive" graph G_S^p, which models the environment, already does not assume that the neighboring robots operate with the same policy. (Rather, it assumes that all possible changes in the

environment are equally probable.) For a swarm of functionally heterogeneous robots, we anticipate the main difficulty to be in defining concurrent sets of desired states for each type of robot, which all combine toward a greater global goal.

- **Including transitions with a probability of failure** An assumption that was taken for the tasks in this work was that, if a robot decides to perform an action, then it will always succeed. In reality, however, this may not always be the case, as certain state–action pairs may have a lower success rate than anticipated in the real world. However, this failure probability can be directly factored in within the transition probabilities of the graphs $G^a_{\mathcal{S}}$ and $G^p_{\mathcal{S}}$. On this note, it could also be possible for a robot to do this online while performing the task.
- **Automatic extraction of desired states** As stated in Sect. 7.2, a limitation of the approach is the need to explicitly define the locally desired states. It can be postulated that this may not always be intuitively done by a swarm designer, particularly as task complexity increases. However, it can be argued that most (if not all) tasks require that some local states are more "desired" than others, and that these must be locally observable by the robot (even if not apparent, or understood, by the designer)—the challenge is to find them. In future work, this challenge could be solved by a multilayered learning approach, whereby a higher-level learner is tasked with extracting the desired local states for a given task.

8 Conclusion

The framework proposed in this paper is based on the idea to microscopically model the local experiences of a robot in a swarm by means of graphs. It then becomes possible to evaluate the PageRank centrality of the local states and use this as a measure of the global behavior of the swarm by evaluating the relative PageRank of local states of interest. This can be used within an optimization procedure in order to optimize the global behavior of swarms while only requiring a local analysis.

This technique provides several advantages which natively include flexibility, robustness, and scalability, which are three key properties of swarm robotics. We have applied it to a pattern formation task, a consensus task, and an aggregation task. The time needed to perform the optimization did not scale with the size of the swarm, which allowed us to tackle swarm sizes that could not be tackled if simulation had been our means of assessment. Moreover, the results obtained were natively independent of global parameters such as the initial starting condition, the environment, or the number of robots in the swarm. The new insights presented in this paper offer a new strategy to evaluate the final behavior of groups from the perspective of the individual. We expect that the approach can generalize to several other tasks, whereby a robot in the swarm and its environment is modeled in analogy to how a user surfs the World Wide Web.

References

Berman, S., Halász, Á., Kumar, V., & Pratt, S. (2007). Algorithms for the analysis and synthesis of a bio-inspired swarm robotic system. In E. Şahin, W. M. Spears, & A. F. T. Winfield (Eds.), *Swarm robotics* (pp. 56–70). Heidelberg: Springer.

Berman, S., Halasz, A., Hsieh, M. A., & Kumar, V. (2009). Optimized stochastic policies for task allocation in swarms of robots. *IEEE Transactions on Robotics*, *25*(4), 927–937.

Berman, S., Nagpal, R., Halász, A. (2011). Optimization of stochastic strategies for spatially inhomogeneous robot swarms: A case study in commercial pollination. In *2011 IEEE/RSJ international conference on intelligent robots and systems* (pp. 3923–3930).

Brin, S., Page, L. (1998). The anatomy of a large-scale hypertextual web search engine. In *Seventh international world-wide web conference (WWW 1998)*.

Busoniu, L., Babuska, R., Schutter, B. D., & Ernst, D. (2010). *Reinforcement learning and dynamic programming using function approximators*. Boca Raton: CRC Press Inc.

Campo, A., & Dorigo, M. (2007). Efficient multi-foraging in swarm robotics. In F. Almeida e Costa, L. M. Rocha, E. Costa, I. Harvey, & A. Coutinho (Eds.), *Advances in artificial life* (pp. 696–705). Berlin: Springer.

Coppola, M., & de Croon, G. C. H. E. (2018). Optimization of swarm behavior assisted by an automatic local proof for a pattern formation task. In M. Dorigo, M. Birattari, C. Blum, A. L. Christensen, A. Reina, & V. Trianni (Eds.), *Swarm intelligence* (pp. 123–134). Cham: Springer.

Coppola, M., Guo, J., Gill, E., & de Croon, G. C. H. E. (2019). Provable self-organizing pattern formation by a swarm of robots with limited knowledge. *Swarm Intelligence*, *13*(1), 59–94.

Correll, N., & Martinoli, A. (2006). Collective inspection of regular structures using a swarm of miniature robots. In M. H. Ang & O. Khatib (Eds.), *Experimental robotics IX: The 9th international symposium on experimental robotics* (pp. 375–386). Heidelberg: Springer.

Correll, N., & Martinoli, A. (2011). Modeling and designing self-organized aggregation in a swarm of miniature robots. *The International Journal of Robotics Research*, *30*(5), 615–626.

Di Mario, E., & Martinoli, A. (2014). Distributed particle swarm optimization for limited-time adaptation with real robots. *Robotica*, *32*(2), 193–208.

Di Mario, E., Navarro, I., Martinoli, A. (2015a). A distributed noise-resistant particle swarm optimization algorithm for high-dimensional multi-robot learning. In *2015 IEEE international conference on robotics and automation (ICRA)* (pp. 5970–5976).

Di Mario, E., Navarro, I., Martinoli, A. (2015b). Distributed particle swarm optimization using optimal computing budget allocation for multi-robot learning. In *2015 IEEE congress on evolutionary computation (CEC)* (pp. 566–572).

Duarte, M., Costa, V., Gomes, J., Rodrigues, T., Silva, F., Oliveira, S. M., et al. (2016). Evolution of collective behaviors for a real swarm of aquatic surface robots. *PLOS One*, *11*, 1–25.

Ericksen, J., Moses, M., Forrest, S. (2017). Automatically evolving a general controller for robot swarms. In *2017 IEEE symposium series on computational intelligence (SSCI)* (pp. 1–8).

Ferrante, E., Duéñez Guzmán, E., Turgut, A. E., Wenseleers, T. (2013). GESwarm: Grammatical evolution for the automatic synthesis of collective behaviors in swarm robotics. In *Proceedings of the 15th annual conference on genetic and evolutionary computation (GECCO)* (pp. 17–24). New York: ACM.

Fornito, A., Zalesky, A., & Bullmore, E. T. (2016). *Centrality and hubs* (pp. 137–161). San Diego: Academic Press.

Francesca, G., & Birattari, M. (2016). Automatic design of robot swarms: Achievements and challenges. *Frontiers in Robotics and AI*, *3*, 29.

Francesca, G., Brambilla, M., Brutschy, A., Garattoni, L., Miletitch, R., Podevijn, G., et al. (2015). Automode-chocolate: Automatic design of control software for robot swarms. *Swarm Intelligence*, *9*(2), 125–152.

Gomes, J., Urbano, P., & Christensen, A. L. (2012). Introducing novelty search in evolutionary swarm robotics. In M. Dorigo, M. Birattari, C. Blum, A. L. Christensen, A. P. Engelbrecht, R. Groß, & T. Stützle (Eds.), *Swarm intelligence* (pp. 85–96). Heidelberg: Springer.

Gomes, J., Urbano, P., & Christensen, A. L. (2013). Evolution of swarm robotics systems with novelty search. *Swarm Intelligence*, *7*(2), 115–144.

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

Hamann, H., & Wörn, H. (2008). A framework of space-time continuous models for algorithm design in swarm robotics. *Swarm Intelligence*, *2*(2), 209–239.

Hamann, H., Valentini, G., Khaluf, Y., & Dorigo, M. (2014). Derivation of a micro-macro link for collective decision-making systems. In T. Bartz-Beielstein, J. Branke, B. Filipič, & J. Smith (Eds.), *Parallel problem solving from nature—PPSN XIII* (pp. 181–190). Cham: Springer.

Hsieh, M. A., Halász, Á., Berman, S., & Kumar, V. (2008). Biologically inspired redistribution of a swarm of robots among multiple sites. *Swarm Intelligence*, *2*(2), 121–141.

Hüttenrauch, M., Šošić, A., Neumann, G. (2017). *Guided deep reinforcement learning for swarm systems*. ArXiv:1709.06011.

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.

Jones, S., Studley, M., Hauert, S., & Winfield, A. (2018). *Evolving behaviour trees for swarm robotics* (pp. 487–501). Cham: Springer.

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

Langville, A. N., & Meyer, C. D. (2006). *Google's PageRank and beyond: The science of search engine rankings*. Princeton: Princeton University Press.

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.

Lerman, K., Martinoli, A., & Galstyan, A. (2005). A review of probabilistic macroscopic models for swarm robotic systems. In E. Şahin & W. M. Spears (Eds.), *Swarm robotics* (pp. 143–152). Heidelberg: Springer.

Martinoli, A., & Easton, K. (2003). Modeling swarm robotic systems. In B. Siciliano & P. Dario (Eds.), *Experimental robotics VIII* (pp. 297–306). Heidelberg: Springer.

Martinoli, A., Easton, K., & Agassounon, W. (2004). Modeling swarm robotic systems: A case study in collaborative distributed manipulation. *The International Journal of Robotics Research*, *23*(4–5), 415–436.

Nelson, A. L., Barlow, G. J., & Doitsidis, L. (2009). Fitness functions in evolutionary robotics: A survey and analysis. *Robotics and Autonomous Systems*, *57*(4), 345–370.

Nolfi, S. (2002). Power and the limits of reactive agents. *Neurocomputing*, *42*(1–4), 119–145.

Page, L., Brin, S., Motwani, R., Winograd, T. (1999). *The PageRank citation ranking: Bringing order to the web*. Technical report 1999-66, Stanford InfoLab.

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.

Reina, A., Miletitch, R., Dorigo, M., & Trianni, V. (2015). A quantitative micro-macro link for collective decisions: The shortest path discovery/selection example. *Swarm Intelligence*, *9*(2), 75–102.

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

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.

Scheper, K. Y. W., Tijmons, S., de Visser, C. C., & de Croon, G. C. H. E. (2016). Behavior trees for evolutionary robotics. *Artificial Life*, *22*(1), 23–48.

Silva, F., Duarte, M., Correia, L., Oliveira, S. M., & Christensen, A. L. (2016). Open issues in evolutionary robotics. *Evolutionary Computation*, *24*(2), 205–236.

Sutton, R. S., Barto, A. G., et al. (1998). *Introduction to reinforcement learning* (Vol. 135). Cambridge: MIT Press Cambridge.

Trianni, V., Nolfi, S., & Dorigo, M. (2006). Cooperative hole avoidance in a swarm-bot. *Robotics and Autonomous Systems*, *54*(2), 97–103.

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.

Yamins, D., Nagpal, R. (2008). Automated global-to-local programming in 1-D spatial multi-agent systems. In *Proceedings of the 7th international joint conference on autonomous agents and multiagent systems (AAMAS)* (vol. 2, pp. 615–622). Richland: International Foundation for Autonomous Agents and Multiagent Systems.

Affiliations

M. Coppola[1,2] · J. Guo[2] · E. Gill[2] · G. C. H. E. de Croon[1]

J. Guo
j.guo@tudelft.nl

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

G. 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