Delft University of Technology

A self-guided approach for navigation in a minimalistic foraging robotic swarm

Adams, Steven; Jarne Ornia, D.; Mazo, Manuel

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

A self-guided approach for navigation in a minimalistic foraging robotic swarm

Steven Adams[1] · Daniel Jarne Ornia[1] · Manuel Mazo Jr[1]

Abstract

We present a biologically inspired design for swarm foraging based on ant's pheromone deployment, where the swarm is assumed to have very restricted capabilities. The robots do not require global or relative position measurements and the swarm is fully decentralized and needs no infrastructure in place. Additionally, the system only requires one-hop communication over the robot network, we do not make any assumptions about the connectivity of the communication graph and the transmission of information and computation is scalable versus the number of agents. This is done by letting the agents in the swarm act as foragers or as guiding agents (beacons). We present experimental results computed for a swarm of Elisa-3 robots on a simulator, and show how the swarm self-organizes to solve a foraging problem over an unknown environment, converging to trajectories around the shortest path, and test the approach on a real swarm of Elisa-3 robots. At last, we discuss the limitations of such a system and propose how the foraging efficiency can be increased.

1 Introduction

In the past thirty years the use of multi-agent techniques to solve robotic tasks has exploded. The advancement of processing power, sensor accuracy and battery sizes has enabled the realisation of coordinated multi-robot problems, where a number of agents organize to solve a specific task. When designing increasingly larger systems, a huge part of methods draw inspiration from biological systems (ants, bees…) where large amounts of agents interact (directly or indirectly) to produce emerging behaviour that results in an optimized solution to a physical problem. Robotic swarms are at the core of these solutions, being decentralised multi-robot systems with biologically inspired minimalist dynamics that where complex behaviours emerge (see e.g. Beni and Wang (1993);

✉ Daniel Jarne Ornia
d.jarneornia@tudelft.nl

Steven Adams
s.j.l.adams@tudelft.nl

Manuel Mazo Jr
m.mazo@tudelft.nl

1 Delft Center for Systems and Control, Delft University of Technology, 2628CD Delft, The Netherlands

Dorigo et al. (2007); Blum and Merkle (2008); Kennedy (2006)).

These biological methods have enabled plenty of theoretical developments, but the applicability of them is still sparse partly due to problem complexities hard to satisfy with minimalistic robots. We focus in this work on the well known foraging problem, defined as the dual problem of exploration/exploitation, where a number of agents start at a given point in space and must find a target in an unknown (possibly time varying) environment, while converging to cycle trajectories that enable them to exploit the found resources as efficiently as possible. Foraging is still a paradigmatic problem when designing very large robotic systems since it combines exploration and on-line path planning. Additionally the duality exploration vs. exploitation is nowadays extremely relevant in Reinforcement Learning and other AI related fields (Thrun, 1992; Ishii et al., 2002; Nair et al., 2018). This problem has been addressed with robotic ant-inspired swarms that use indirect communication through some "pheromone" (either virtual or chemical) (Fujisawa et al., 2008; Russell, 1997; Johansson and Saffiotti, 2009). Some early work was done in Drogoul and Ferber (1993) showing how robots can use pheromone based information to explore and collect. After that, authors in e.g., Sugawara et al. (2004); Fujisawa et al. (2014); Campo et al. (2010);

Alers et al. ([2014](#)); Font Llenas et al. ([2018](#)); Mayet et al. ([2010](#)); Garnier et al. ([2007](#)); Ziparo et al. ([2007](#)); Ducatelle et al. ([2009](#)); Svennebring and Koenig ([2004](#)) have explored several practical methods to implement pheromone-based robotic navigation. However, complexities explode when designing very large multi-robot swarms, be it in terms of computation, data transmission or dynamic coupling. Additionally, these systems often include restrictive assumptions in terms of e.g., sensor range, memory storage in the agents, computational capacity or reliability of communications.

1.1 Related work

There have been multiple proposals for de-centralised multi robot foraging systems. Authors in Payton et al. ([2001](#)) propose a network (multi-hop) algorithm that propagates counter signals through a grid of robots, such that the robots accumulate around the shortest path between two targets in space, propagating messages using line-of-sight communication. The authors in Ducatelle et al. ([2011b](#)) present an navigational algorithm for a homogeneous swarm to travel between two targets in space that requires the agents to flood the network with their believe of the relative locations of each unique identifiable agent, and assumes a pre-positioned agent at the target locations.

In Nouyan et al. ([2008](#)) a path planning algorithm is presented that explores the environment using chains of sweeping robots, or grow a grid of robots storing a vector field pointing towards a given target relying on relative positioning measures and one-hop line-of-sight communication via LED signalling. In Hrolenok et al. ([2010](#)); Russell et al. ([2015](#)), authors equip agents with deployable beacon devices to guide navigation and store pheromone values. Authors treat pheromones as utility estimates for environmental states, and agents rely on line-of-sight one-hop communication and a relative position measure to the beacons to generate foraging trajectories. In Ducatelle et al. ([2011a](#)) the collaboration between two different swarms is investigated, using foraging as a paradigmatic problem, having a pre-deployed swarm able to overlook the full domain and act as guiding agents for the second swarm to solve the foraging problem. In this case the use of vectors stored in the guiding swarm is explored, with such vectors being updated using the relative positions of the foraging swarm. In Reina et al. ([2017](#)); Talamali et al. ([2020](#)) the authors use a virtual-reality approach to implement the pheromone field, allowing the robots to have access to this virtual pheromone from a central controller, enabling effective foraging.

In Lemmens et al. ([2007](#)); Lemmens and Tuyls ([2009](#)), authors propose bee-inspired path integration algorithms on a computational set-up, where agents use landmarks to store pheromone-based information when a change in direction is required.

Note that (Nouyan et al., [2008](#); Ducatelle et al., [2011b](#); Payton et al., [2001](#)) assume agents in the swarm communicate directly with other agents, while (Hrolenok et al., [2010](#); Russell et al., [2015](#); Ducatelle et al., [2011a](#)) de-couple this and propose an environment-based interactions where agents only write and read data into locally reachable beacons. However, all assume knowledge of relative positions between agents.

1.2 Main proposal

We aim in this work to provide a *scalable* and *robust* solution to the problem of decentralised foraging robots, in terms of communication disruptions, robot failures and changing environment. We propose a design a minimalistic swarm system capable of solving the foraging problem by using a form of pheromone-inspired communication with the following restrictions:

- All agents do not have knowledge of relative (or global) positions with respect to other robots, and only need an orientation measure (a compass).
- The system relies on one-hop communication only with limited range, and does not require direction or distance of signals nor line-of-sight connectivity.
- The system is fully distributed and needs no infrastructure in place.

The contribution is therefore split in two main branches:

1. Swarm doesn't require position measurements, multi-hop or line of sight communication.
2. Swarm is robust versus *changing environments*, *communication disruptions* and *agent failures*, demonstrated through extensive simulations.

The goal of our proposal is to obtain a swarm system that is deployable in real life, without knowledge of the specific environment and that adapts to changing environments and unforeseen events. This goal dictates contributions 1 and 2 above. In this way, contribution 1 is an improvement with respect to, among others, Hrolenok et al. ([2010](#)); Russell et al. ([2015](#)); Ducatelle et al. ([2011a](#), [b](#)); Nouyan et al. ([2008](#)); Sperati et al. ([2011](#)); Hoff et al. ([2013](#)). Although there are (few) examples of foraging swarm work that considers changing environments, robustness studies in the form of contribution 2 do not appear in detail in existing literature (with the exception of Nouyan et al. ([2008](#)) where a swarm path formation problem is considered) for a fully distributed and autonomous foraging swarm, and we believe it is of great importance for a swarm intended to operate in the real world.

1.3 Preliminaries

We use calligraphic letters for sets ($\mathcal{A}$), regular letters for scalars ($a \in \mathbb{R}$) and bold letters for vectors ($\mathbf{a} \in \mathbb{R}^n$). We consider discrete time dynamics $k \in \mathbb{N}$, and we define an inter-sampling time $\tau \in \mathbb{R}_+$ such that we keep a "total" time measure $t = \tau k$. With vectors we use $\|\mathbf{v}\|$ as the euclidean norm, and $\langle \mathbf{v} \rangle := \frac{\mathbf{v}}{\|\mathbf{v}\|}$. We use the diagonal operator $D(\cdot) = \mathrm{diag}(\cdot)$.

2 Problem description

Take a swarm of N agents $\mathcal{A} = \{1, 2, \ldots, N\}$ navigating in a bounded domain $\mathcal{D} \subset \mathbb{R}^2$, where $\mathcal{D}$ is compact and connected (possibly non-convex). We define $\mathbf{x}_a(k) \in \mathcal{D}$ as the position of agent a at (discrete) time k, and velocity $\mathbf{v}_a(k) = v_0(\cos(\alpha_a(k)) \ \sin(\alpha_a(k)))^T$ with $\alpha_a(k) \in [-\pi, \pi)$ as its orientation. We define the dynamics of the system to be in discrete time, such that the positions of the agents evolve as

$$\mathbf{x}_a(k + 1) = \mathbf{x}_a(k) + \mathbf{v}_a(k)\tau. \tag{1}$$

Consider the case where the swarm is trying to solve a *foraging problem*.

Definition 1 (Foraging problem). A foraging problem on an unknown domain $\mathcal{D}$ is the joint problem of finding a target region $\mathcal{T} \subset \mathcal{D}$ when starting on a different region $\mathcal{S} \subset \mathcal{D}, \mathcal{S} \cap \mathcal{T} = \emptyset$, and eventually following (sub) optimal trajectories.[1]

The main goal when solving a foraging problem is to complete trajectories between $\mathcal{S}$ and $\mathcal{T}$ as fast as possible (through the shortest path), back and forth. Then, the quality of the solutions can be quantified by the amount of trips between the $\mathcal{S}$ and $\mathcal{T}$ performed by all agents in some time interval. To design such a swarm, we make the following assumptions on the swarm agents' capabilities.

Assumption 1 (Swarm constraints[2])

1. Agents have a small memory, enough to store scalars and vectors in $\mathbb{R}^2$, and enough computational power to perform sums and products of vectors.
2. Agents have the ability to send and receive basic signals (up to 6 scalar values), within a maximum range δ, every τ seconds.

3. Agents have some form of collision avoidance mechanism, acting independently of the design dynamics.
4. Agents have sensing ability to detect $\mathcal{S}, \mathcal{T}$.
5. Agents have some measure of angular orientation (e.g., a compass).
6. Agents are able to remain static.

Observe that we do not assume the ability to measure directionality in the signals, nor any form of self-localisation capacity. Additionally, the agents do not have access to any form of global information about $\mathcal{D}$, do not have unique identifiers and do not require line-of-sight interactions. The swarm does not require either any form of infrastructure in place. At last, the swarm relies on one-hop communication only, and we do not require the communication to be synchronous. That is, communication happens on an agent-to-agent basis, and agents do not cascade information through the communication network. The challenge to be solved in this work is then the following.

problem 1 *Design a swarm of N agents that solves a foraging problem over an unknown domain $\mathcal{D}$, while satisfying the set of Assumptions 1, and does so with guarantees.*

3 Proposal: self guided swarm

We now present our design for a self-guided swarm that solves the foraging problem presented in Sect. 2. Our design is based on the idea of allowing agents in the swarm to behave as "beacons" (agents in charge of guiding others) or "foragers" (agents in charge of travelling from $\mathcal{S}$ to $\mathcal{T}$). Beacon agents store weight values and guiding velocity vectors, which they broadcast to the foragers in the swarm to generate foraging trajectories. We first describe the different modes the agents can operate in, the switching rules between modes, and then the dynamics in every mode.

3.1 States and transitions

Let us then split the swarm into three groups of agents. We will use the state variable $s_a(k) \in \{B, F_1, F_2\}$ with $a \in \mathcal{A}$ to indicate:

1. $s_a(k) = B \Rightarrow a$ is a beacon.
2. $s_a(k) = F_1 \Rightarrow a$ is a forager looking for $\mathcal{T}$.
3. $s_a(k) = F_2 \Rightarrow a$ is a forager looking for $\mathcal{S}$.

We use $s = F_1, F_2$ as the different foraging states. Then, we can group the agents in time-dependent sub-sets: the beacons $\mathcal{B}(k) : \{a \in \mathcal{A} : s_a(k) = B\}$ and the foragers $\mathcal{A}^s(k) := \{a \in \mathcal{A} : s_a(k) = s\}$.

---

[1] A trajectory is optimal if there exists no other trajectory connecting $\mathcal{S}$ and $\mathcal{T}$ that has a shorter length. We say (informally) it is sub-optimal if it is relatively close (in travel time) to the optimal trajectory length between $\mathcal{S}$ and $\mathcal{T}$.

[2] Many off-the-shelf small robotic systems satisfy these assumptions, e.g. Elisa-3, e-Puck, Sphero Bolt….

At $t = 0$ all agents are initialised at $\mathcal{S}$. One agent is chosen to be the initial beacon, and all others are initialised as foragers looking for $\mathcal{T}$:

$$
\begin{aligned}
&\mathbf{x}_a(0) \in \mathcal{S} \; \forall a \in \mathcal{A}, \\
&|\mathcal{B}(0)| = 1, \\
&|\mathcal{A}^{F_1}(0)| = N - 1.
\end{aligned}
\tag{2}
$$

This initial beacon can be chosen at random, or based on some order of deployment of the swarm. Let us now define the regions of influence of every agent as $\mathcal{D}_a(k) := \{\mathbf{x} \in \mathcal{D} : \|\mathbf{x} - \mathbf{x}_a(k)\|_2 \leq \delta\}$, for some maximum instrument range $\delta \in \mathbb{R}_+$. As time evolves, the agents switch between states following the logic rules $\forall a \in \mathcal{A}$:

$$
s_a(k+1) = \begin{cases}
B & \text{if } \nexists\, b \in \mathcal{B}(k) : \mathbf{x}_b(k) \in \mathcal{D}_a(k), \\
F_1 & \text{if } s_a(k) = F_2 \wedge \mathbf{x}_a(k) \in \mathcal{S}, \\
F_2 & \text{if } s_a(k) = F_1 \wedge \mathbf{x}_a(k) \in \mathcal{T}, \\
s_a(k) & \text{else,}
\end{cases}
\tag{3}
$$

The switching rule in (3) is interpreted in the following way. If a forager moves to a point in the domain where there is no other beacons around, it becomes a beacon. If a forager is looking for the set $\mathcal{T}$ (mode 1) and finds it, it switches to finding the starting set $\mathcal{S}$ (mode 2), and the other way around. For now we do not consider transitions from beacon to forager.

3.2 Dynamics

We assume that beacons remain static while in beacon state:

$$
\mathbf{v}_b(k) = \mathbf{0}, \quad \mathbf{x}_b(k) = \mathbf{x}_b(k_b) \; \forall k \geq k_b,
\tag{4}
$$

where k_b is the time step when agent b switched to beacon state. Beacon agents store weight values $\omega_b^s(k) \in \mathbb{R}_+$ and guiding velocity vectors $\mathbf{u}_b^s(k) \in \mathbb{R}^2$, initialised at zero for all agents in the swarm. At every time-step, beacon agents broadcast their stored values $\omega_b^s(k), \mathbf{u}_b^s(k)$ with a signal of radius δ. Let us define the set of neighbouring beacons to a forager in state s, $f \in \mathcal{A}^s(k)$ as $\mathcal{B}_f(k) := \{b \in \mathcal{B}(k) : \mathbf{x}_b(k) \in \mathcal{D}_f(k)\}$, and the set of neighbouring foragers to a beacon $b \in \mathcal{B}(k)$ as $\mathcal{F}_b^s(k) := \{f \in \mathcal{A}^s(k) : \mathbf{x}_f(k) \in \mathcal{D}_b(k)\}$.

At every time-step each forager receives a set of signals from neighbouring beacons, and computes a reward function $\Delta_f^s(k) \in \mathbb{R}_+$:

$$
\Delta_f^s(k) = \gamma_f^s(k) + \lambda \max_{b \in \mathcal{B}_f(k)} \omega_b^s(k),
\tag{5}
$$

where $\lambda \in [0, 1]$ is a diffusion rate, and for some $r \in \mathbb{R}_+$,

$$
\gamma_f^s(k) = \begin{cases}
r & \text{if } s_f(k) = F_1 \wedge \mathbf{x}_f(k) \in \mathcal{S}, \\
r & \text{if } s_f(k) = F_2 \wedge \mathbf{x}_f(k) \in \mathcal{T}, \\
0 & \text{else.}
\end{cases}
\tag{6}
$$

The reward function in (5) works as follows: Foragers listen for weight signals from neighbouring beacons and broadcast back the maximum discounted weight, plus a constant reward if they are in the regions $\mathcal{S}$ or $\mathcal{T}$ depending on their state. The discounted term λ serves the role of a hop-count in other literature examples. However, in our case the robots do not need to keep count of messages, nor cascade messages through the network. This propagation of information through the network is done in a fully sparse and memory-less way (since robots only listen for messages, compute new weights and broadcast messages to robots around them), without the need of keeping connectivity in the beacon network. Observe that (5) depends on s, indicating that foragers listen and reinforce only the weights corresponding to their internal state value. The beacons update their weight values using a (possibly different) discount factor $\rho \in [0, 1]$ as

$$
\omega_b^s(k+1) = (1-\rho)\omega_b^s(k) + \rho \frac{\sum_{f \in \mathcal{F}_b^s(k)} \Delta_f^s(k)}{|\mathcal{F}_b^s(k)|}.
\tag{7}
$$

The iteration in (7) is only applied if there are indeed neighbouring foragers around a beacon, so $|\mathcal{F}_b^s(k)| \geq 1$. Otherwise, $\omega_b^s(k+1) = \omega_b^s(k)$.

The update rule of $\mathbf{u}_b^s(k)$ is similarly:

$$
\mathbf{u}_b^s(k+1) = (1-\rho)\mathbf{u}_b^s(k) - \rho \frac{\sum_{f \in \mathcal{F}_b^s(k)} \mathbf{v}_f(k)}{|\mathcal{F}_b^s(k)|}.
\tag{8}
$$

and again, $|\mathcal{F}_b^s(k)| = 0 \Rightarrow \mathbf{u}_b^s(k+1) = \mathbf{u}_b^s(k)$. At the same time that beacons update their stored weight values based on the foragers around them, they update as well the guiding velocity vectors by adding the corresponding velocities of the foragers around them (with an opposite sign). The logic behind this has to do with the reward function in (5). Foragers looking for $\mathcal{T}$ update weights and guiding velocities associated with state F_1, but to indicate that they are in fact moving out of $\mathcal{S}$, we want to update the guiding velocities based on the opposite direction that they are following.

Until now we have defined the dynamics of the beacon agents: position and velocities in (4) and update rules for $\omega_b^s(k)$, $\mathbf{u}_b^s(k)$ in (7) and (8). We have yet to define the dynamics of the foragers. At every step, the foragers listen for guiding velocity and weight signals from beacons around them. With this information, they compute the guiding vec-

tor:

$$\hat{\mathbf{v}}_f^s(k) := v_0 \left\langle \sum_{b \in \mathcal{B}_f(k)} \omega_b^{\bar{s}}(k) \mathbf{u}_b^{\bar{s}}(k) \right\rangle, \tag{9}$$

where $\bar{s}$ is the opposite state to s, $s = F_1 \iff \bar{s} = F_2$. At every time-step foragers choose stochastically, for a design exploration rate $\varepsilon \in (0, 1)$, if they follow the guiding vector $\hat{\mathbf{v}}_f^s(k)$ or they introduce some random noise to their movement. Let α_u be a random variable taking values $(-\pi, \pi]$, following some probability density function $p(\alpha_u)$, and let $\tilde{\alpha}_a(k) := \alpha_a(k) + \alpha_u$. Then, $\forall f \in \mathcal{A}^s(k)$:

$$\begin{aligned} &\Pr\left\{\mathbf{v}_f(k+1) = v_0 \begin{pmatrix} \cos(\tilde{\alpha}_a(k)) \\ \sin(\tilde{\alpha}_a(k)) \end{pmatrix}\right\} = \varepsilon, \\ &\Pr\{\mathbf{v}_f(k+1) = \hat{\mathbf{v}}_f^s(k)\} = 1 - \varepsilon. \end{aligned} \tag{10}$$

Additionally, we add a fixed condition for an agent to turn around when switching between foraging states. That is,

$$\mathbf{v}_f(k+1) = -\mathbf{v}_f(k) \quad \text{if } s_f(k+1) \neq s_f(k). \tag{11}$$

With (10) and (11) the dynamics of the foragers are defined too. We have experimentally verified that, over a diverse variety of set-ups, the weights $\omega_b^s(k)$ converge on average to a fixed point forming a gradient outwards from $\mathcal{S}$ and $\mathcal{T}$, therefore guiding the swarm to and from the goal regions.

3.3 Implementation

The dynamics presented result in the following algorithms running in the robots. The robots switch between Algorithms 1 and 2, depending on if they are beacons or foragers.

Algorithm 1 Behaviour of beacons
---
while $s_b(k) = 0$ **do**
    Broadcast $\omega_b^s(k)$, $\mathbf{u}_b^s(k)$;
    Listen for signals during τ seconds;
    Compute $\omega_b^s(k+1)$, $\mathbf{u}_b^s(k+1)$;
    Move according to $\mathbf{v}_b(k+1)$;
    if *Obstacle* **then**
        Move to avoid obstacle;
    end if
    Check transitions in (3);
end while

---

4 Results and guarantees

To show that the swarm finds the target region $\mathcal{T}$ and eventually converges to close to optimal trajectories between $\mathcal{S}$ and $\mathcal{T}$, let us first state the assumptions for the following theoretical results.

Algorithm 2 Behaviour of foragers
---
while $s_f(k) \neq 0$ **do**
    Listen for signals during τ seconds;
    Broadcast $\mathbf{v}_f(k)$, $\Delta_f^s(k)$;
    Compute $\mathbf{v}_f(k+1)$ from (10) and (11);
    Move according to $\mathbf{v}_f(k+1)$;
                                               ▷ Asynchronous action
    if *Obstacle* **then**
        Move to avoid obstacle;
    end if
    Check transitions in (3);
end while

Assumption 2

1. Any $\mathcal{D}_k \subseteq \mathcal{D}$ is a compact disc of radius δ.
2. $\tau < \frac{\delta}{2v_0}$, and can be chosen small enough in comparison to the diameter of the domain $\mathcal{D}$.
3. The regions $\mathcal{S}$ and $\mathcal{T}$ are compact discs of radius, at least, $2\tau v_0$.

4.1 Domain exploration

Remark 1 It holds by construction that $\exists b : \mathbf{x}_a(k) \in \mathcal{D}_b(k) \ \forall k, \ \forall a \in \mathcal{A}^s(k)$.

From the transition rule in (3) it follows that, whenever $\mathbf{x}_a(k) \notin \mathcal{D}_b(k)$ for any beacon $b \in \mathcal{B}(k)$, it becomes a beacon, therefore covering a new sub-region of the space. To obtain the first results regarding the exploration of the domain, we can take inspirations from results in RRT exploration (Kuffner & LaValle, 2000), where the agents perform a similar fixed length step exploration over a continuous domain. Let $p_a(\mathbf{x}, k)$ be the agent probability density of point $\mathbf{x}$ at time k. We obtain the following results.

Proposition 1 *Let $\mathcal{D} \subset \mathbb{R}^2$ be convex. Let some $a \in \mathcal{A}$ have $\mathbf{x}_a(k_0) = \mathbf{x}_0$. Then, for any convex region $\mathcal{D}_n \subset \mathcal{D}$ with non-zero Lebesgue measure, there exists $\kappa_n \in \mathbb{R}_+$ and time $k_n \in \mathbb{N}$ such that*

$$\Pr[\mathbf{x}_a(k_n + k_0) \in \mathcal{D}_n \mid \mathbf{x}_a(k_0)] \geq \varepsilon^{k_n} \kappa_n. \tag{12}$$

Proof (Sketch of proof) At every time-step k, with probability ε, it holds $\alpha_a(k+1) = \alpha_a(k) + \alpha_u$, and observe that $\alpha_a(k+1) \in (-\pi, \pi]$. Let agent a be at point $\mathbf{x}_0$ at time k_0, and take the event where for every time-step after k_0 the agent always chooses the random velocity. For $k = k_0 + 1$, the set of points $\mathcal{X}_a(1) \subseteq \mathcal{D}$ satisfying $p_a(k_0 + 1, \mathbf{x}) > 0$ is $\mathcal{X}_a(k_0 + 1) = \{x \in \mathcal{D} : \|x - \mathbf{x}_0\|_2 = v_0\tau\}$. One can verify that $\mathcal{X}_a(k_0 + 1)$ forms a circle in $\mathbb{R}^2$ around $\mathbf{x}_0$. Now for $k = k_0 + 2$, the set $\mathcal{X}_a(k_0 + 2)$ is

$$\begin{aligned} \mathcal{X}_a(k_0 + 2) = \{\mathbf{x} \in \mathcal{D} : &\|\mathbf{x} - \mathbf{x}_1\|_2 = \tau v_0, \\ &\mathbf{x}_1 \in \mathcal{X}_a(k_0 + 1)\}. \end{aligned} \tag{13}$$

In this case, the set in (13) forms a ball of radius $2v_0\tau$ around $\mathbf{x}_0$. At $k = k_0 + 2$, it holds from (13) that $p_a(\mathbf{x}, k_0 + 2) > 0 \,\forall\, \mathbf{x} \in \mathcal{X}_a(2)$. Then, for any subset $\mathcal{D}_2 \subseteq \mathcal{X}_a(2)$, it holds that $\Pr[\mathbf{x}_a(2) \in \mathcal{D}_2 \mid \mathbf{x}_a(k_0)] \geq \varepsilon^2 \kappa_2$, where $\kappa_2 \in \mathbb{R}_+$ is a function of the set $\mathcal{D}_2$ and the probability density $p_a(\mathbf{x}, k_0 + 2)$. The sets $\mathcal{X}_a(k)$ are balls centred at $\mathbf{x}_0$ and radius $k\tau$, and $p_a(\mathbf{x}, k) > 0 \,\forall\, x \in \mathcal{X}_a(k)$. Let at last $\mathcal{D}_n$ be any convex subset of $\mathcal{D}$ with non-zero Lebesgue measure, and $k_n = \min\{k : \mathcal{D}_n \subset \mathcal{X}_a(k)\}$. Then, $\Pr[\mathbf{x}_a(k_0 + k_n) \in \mathcal{D}_n \mid \mathbf{x}_a(k_0)] \geq \varepsilon^{k_n} \kappa_n$ for some $\kappa_n > 0$.

□

Now we can draw a similar conclusion for the case where $\mathcal{D}$ is non-convex.

Lemma 1 *Let $\mathcal{D}$ be a non-convex path-connected domain. Let some $a \in \mathcal{A}$ have $\mathbf{x}_a(k_0) = \mathbf{x}_0$. Then, for any convex region $\mathcal{D}_n \subset \mathcal{D}$ of non-zero Lebesgue measure, there exists $\tau > 0$ and $\kappa_n > 0$ such that we can find a finite horizon $k_n \in \mathbb{N}$:*

$$\Pr[\mathbf{x}_a(k_n + k_0) \in \mathcal{D}_n \mid \mathbf{x}_a(k_0)] \geq \varepsilon^{k_n} \kappa_n. \tag{14}$$

Proof (Sketch of proof) If $\mathcal{D}$ is connected, then for any two points $\mathbf{x}_0, \mathbf{x}_n \in \mathcal{D}$, we can construct a sequence of balls $\{\mathcal{X}_0, \mathcal{X}_1, \ldots, \mathcal{X}_{k_n}\}$ of radius $R \geq v_0\tau$ centred at $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{k_n}$ such that the intersections $\mathcal{X}_i \cap \mathcal{X}_{i+1} \neq \emptyset$ and are open sets, and $\mathbf{x}_i \in \mathcal{X}_{i-1}$. Then, we can pick τ to be small enough such that every ball $\mathcal{X}_i \subset \mathcal{D}$ does not intersect with the boundary of $\mathcal{D}$, and we can apply now Proposition 1 recursively at every ball. If $\|\mathbf{x}_i - \mathbf{x}_{i-1}\|_2 < 2v_0\tau$, then from Proposition 1 we know $p(\mathbf{x}_i, k_{i-1} + 2) > 0$ since, for a given $\mathbf{x}_{i-1}$ and k_{i-1}, any point $\mathbf{x}_i$ has a non-zero probability density in at most 2 steps. Then, it holds that $p(\mathbf{x}_n, k_{n-1} + 2) > 0$ for some $k_{n-1} \in [k_n, 2k_n]$, and for a target region $\mathcal{D}_n$: $\Pr[\mathbf{x}_a(k_n + k_0) \in \mathcal{D}_n \mid \mathbf{x}_a(k_0)] \geq \varepsilon^{k_n} \int_{\mathcal{D}_n} p_a(\mathbf{x}_n, k_n + k_0) d\mathbf{x} \geq \varepsilon^{k_n} \kappa_n$ for some $\kappa_n > 0$. □

It follows from Lemma 1 that for a finite domain $\mathcal{D}$ every forager agent visits every region infinitely often. For a given initial combination of foraging agents, we have now guarantees that the entire domain is be explored and covered by beacons as $k \to \infty$.

4.2 Foraging

We leave for future work the formal guarantees regarding the expected weight field values $\omega_b^s(k)$ and guiding velocities $\mathbf{u}_b^s(k)$. Based on existing literature (Payton et al., 2001; Jarne Ornia & Mazo, 2020), one could model the network of beacons as a discrete graph with a stochastic vertex weight reinforcement process based on the movement of the robots to proof convergence of both $\omega_b^s(k)$ and $\mathbf{u}_b^s(k)$. Such results would also allow us to study the limiting distribution of the

robots across the space, as well as their trajectories. In this work we have experimentally verified that, over a diverse variety of set-ups, the weights $\omega_b^s(k)$ converge on average to a fixed point forming a gradient outwards from $\mathcal{S}$ and $\mathcal{T}$, therefore guiding the swarm to and from the goal regions.

5 Experiments

For the experiments we present an extensive statistical analysis of simulations running on *Webots* (Michel, 2004), adaptability results simulated in a particle swarm simulator and an application running on real Elisa-3 robot (GCTronic). We used the *Webots* simulator for the implementation of the work given its capabilities to simulate realistic robots in physical environments, and since it includes Elisa-3 robot models which satisfy the restrictions in Assumption 1 (using odometry for direction of motion measures). The swarm agents are able to listen almost-continuously, store any incoming signals in a buffer, and empty the buffer every τ seconds. The maximum range δ is virtually limited to restrict the communication range and simulate larger environments where not all robots are capable of reaching each-other. Additionally, the robots use a built-in collision avoidance mechanism, triggered when the infra-red sensors detect an obstacle closer than 2 cm. See Appendix 4 for a detailed description of the collision avoidance mechanism.

The final parameters used for both the simulation and the real swarm are presented in Table 1. For details on how these parameters were selected and more information on how they affect the system, see Appendix 1. To fully evaluate the performance of the method we now include results regarding convergence and efficacy metrics. We are interested in measuring two main properties in such swarm:

1. Foraging performance over finite horizon $t \in [t_{\text{hit}}, T]$: *Navigation delay $d(t_{hit}, T)$* (Ducatelle et al., 2011b).

$$d(t_{\text{hit}}, T) = \frac{1}{|\mathcal{A}|} \sum_{i \in \mathcal{A}} \frac{T - t_{\text{hit}}}{\#\text{trips}_i}, \tag{15}$$

   where t_{hit} is defined as the first time step an agent reaches the target region.
2. Accumulation of agents around foraging trajectories: *Hyerarchic social entropy $S(\mathcal{A})$*.

For conciseness in the results we will address in this section the performance in terms of navigation delay. For further details on the entropy metric and results see Appendix 2. The experiment section is split in three parts:

Table 1 Simulation parameters

| Parameters							
ρ	λ	r	ε	$\tau\,(s)$	$\delta\,(m)$	$v_0(\frac{m}{s})$	σ^2
0.01	0.8	1	0.05	1	0.4	0.25	0.01

1. *Foraging results* We perform a statistical analysis of the foraging performance in terms of the metrics above for different swarm sizes.
2. *Swarm adaptability* We present the results obtained for the cases with mobile obstacles, communication errors and measurement errors.
3. *Real swarm* We demonstrate an example of a real robotic swarm solving a foraging problem.

5.1 Foraging results

We simulated a Webots swarm in an environment of size 2.5 m × 3 m, with and without obstacles, where the nest and food are 2 m apart, to evaluate the influence of the number of agents N in a foraging benchmark scenario. All simulations are run for horizon $T = 400$ s, since that was observed to be enough for the swarm to stabilize the trajectories. A first result on the swarm configuration under these parameters is shown in Fig. 1.

The robots are released in batches to avoid that the starting regions is overcrowded and the movements of the agents in the densely populated region are solely determined by their collision avoidance mechanism, which is called the *overcrowding effect*. Fig. 1a and b are simplified visualizations of the Webots simulator, The size of green dots represents relative amount of weights stored at the beacons, and the plotted vectors are the guiding velocities $\mathbf{u}^s(k)$ at each beacon. One can see how the swarm is able to navigate to the target set, and does so around the shortest path. Note however how the random motion still affects some members of the swarm, and how the agents' accumulation seems to be restricted by how well the beacons represent the desired velocity field. This can be clearly seen in Fig. 1. Most swarm members are indeed moving back and forth from $\mathcal{S}$ to $\mathcal{T}$, but do so spreading between the rows of beacons closer to the minimum length path.

In Tables 2 and 3 we present the navigation delay for three different sizes of the swarm, $N = 49$, $N = 81$ and $N = 157$. These numbers were picked as follows. First, $N = 49$ is a 7×7 group of robots. With the selected $\delta = 0.4$ m, we can expect it will take between 30 and 40 beacons to cover the entire area (assuming every robot covers $\approx \frac{\pi 0.4^2}{2}$ m^2 of area). This leaves a group of 10–20 robots for foraging tasks, which can be considered the minimum to obtain significant results. Second, $N = 81$ is a 9×9 group of robots, which

(a) WebotsSwarm State at 50s

(b) Webots Swarm State at 300s
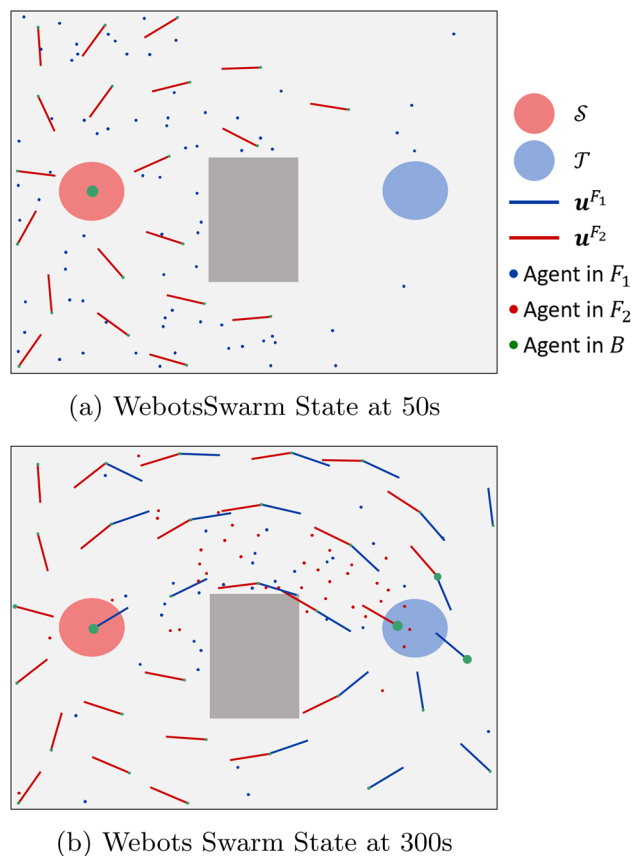
Fig. 1 Webots swarm state with obstacle

Table 2 Navigation delay for $[t_{hit}, T]$ without obstacle

Swarm size	Foragers		Random	
	Mean	Std	Mean	Std
49	**17.45**	5.46	**294.84**	12.53
81	**16.45**	0.92	**322.90**	41.38
157	**103.3**	63.44	**1152.03**	709.4

Bold values indicate the mean of the results

Table 3 Navigation delay for $[t_{hit}, T]$ with symmetric obstacle

Swarm size	Foragers		Random	
	Mean	Std	Mean	Std
49	**30.18**	7.42	**375.74**	27.85
81	**23.55**	1.53	**433.93**	66.30
157	**97.54**	22.25	**1250.24**	706.13

Bold values indicate the mean of the results

was observed in the first experiments to perform very efficiently for the selected environment. At last, $N = 157$ is the maximum robots Webots could run simultaneously (in our equipment), and is enough to produce the *overcrowding effect*.

Each value includes results from 12 independent test runs. The navigation delay is presented for 2 different scenarios:

1. *Random* Agents moving at random.
2. *Foragers* Only forager agents $N - \mathcal{B}(T)$.

Computing the navigation delay for the Foragers case is equivalent to using $d(t_{\text{hit}}, T) = \frac{1}{|N - \mathcal{B}(T)|} \sum_{i \in \mathcal{A}} \frac{T - t_0}{\#\text{trips}_i}$, such that the number of trips is averaged among the number of foragers. We also include an absolute lower bound, corresponding to the absolute minimum possible travel time in the considered scenario. We can now extract several conclusions with respect to the swarm sizes. For a size of $N = 49$ or smaller too many agents are needed as beacons, hence the performance (specially when considering the full swarm) is significantly worse than for bigger swarms. When increasing the size of the swarm to $N = 81$ the performance increases and the variance in the results is reduced, but this tendency reverses for too large swarms due to the *overcrowding* effect, as it can be seen in the results for $N = 157$. Another point worth noting is that the lower bound of navigation delay where robots know and follow the optimal trajectory to the target is $\approx 8s$ without obstacles and $\approx 9s$ with obstacle. These conclusions with respect to the swarm size are subject to mainly two factors: The size of the environment and the radius of influence δ. For larger environments and lower δ we can expect a scaling of the problem, such that we need more robots to solve the foraging task and it will also take more robots to reach the overcrowding point.

5.2 Swarm adaptability

The ultimate goal of the system proposed in this work is to obtain swarms as efficiently as possible, without requiring complex single systems, with high flexibility and robustness. In this line of thought, we want to analyse the adaptability of the swarm with respect to three points:

1. Removing beacons and allowing them to move.
2. Moving obstacles (non-stationary environments).
3. Agent failure and measurement errors.

For all the scenarios, we simulate a baseline swarm with $N = 101$ agents and the parameters as in Table 2, and every numerical result includes a sample of 50 simulations.

5.2.1 Movable beacons

To increase overall efficiency of the system we implement controllers in the beacons to allow them to *turn back to foragers* when their weights have not been updated for a long time and to *move to more transited areas* of the environment. This ensures only the necessary amount of beacons
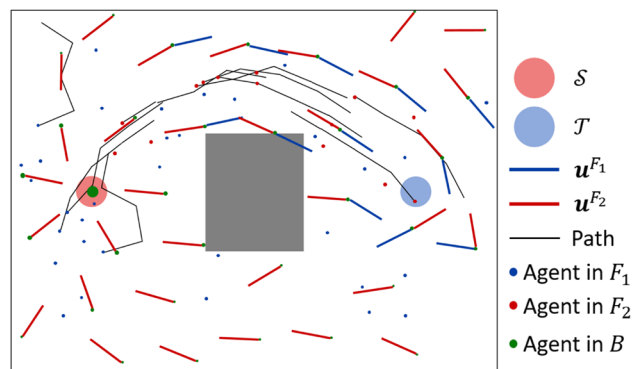
(a) Adapting swarm after $100s$.
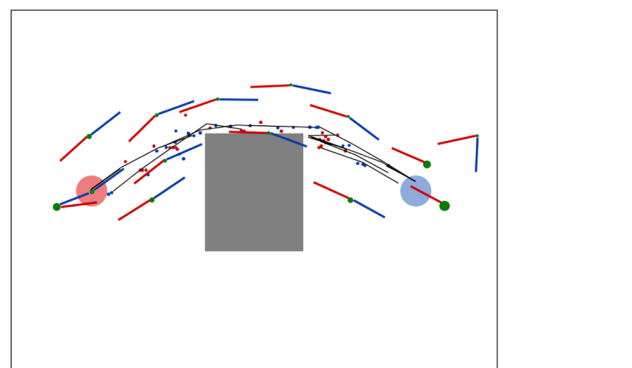
(b) Adapting swarm after $200s$.
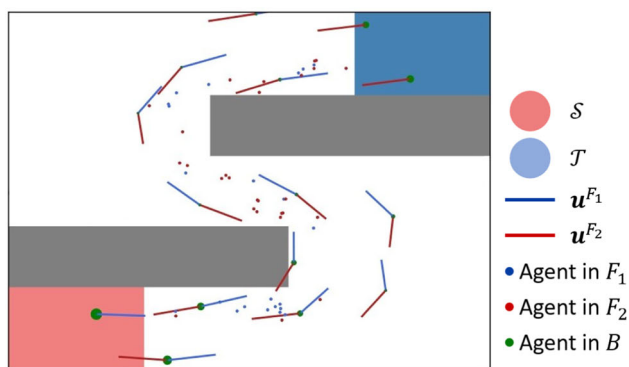
Fig. 2 Swarm with movable beacons

are employed as the system evolves, and increases the granularity (or the definition) of the paths that are being used more often, possibly enabling more optimal configurations.

We implemented such method allowing beacons to turn to foragers when their weights are lower than a specific threshold for too long, and adding a P controller to the velocity of the beacons (set to 0 in previous experiments):
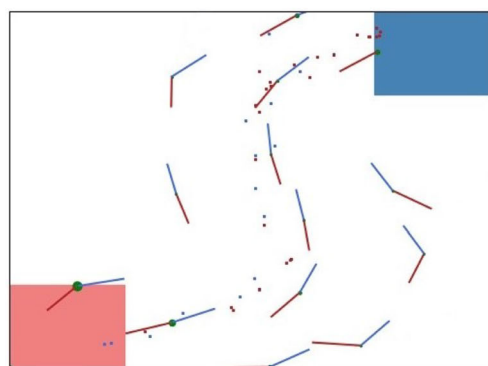
$$v_b(k) = \begin{cases} 0 & \text{if } \mathcal{D}_b(k) \cap \mathcal{S} \cap \mathcal{T} \neq \emptyset, \\ v_{b,0} \langle \sum_s \mathbf{u}_b^s(k) \rangle & \text{else} \end{cases} \quad (16)$$

with $v_{b,0} \in \mathbb{R}_+$, $v_{b,0} << v_0$ the beacons speed. This controller allows the beacons to slowly move towards the mid direction between their two guiding velocity vectors $\mathbf{u}_b^s(k)$. The logic between this controllers is straight-forward: If the most optimal trajectory is a straight line, the most optimal configuration for the beacons would be to sit along the line, with the guiding vectors on 180°.
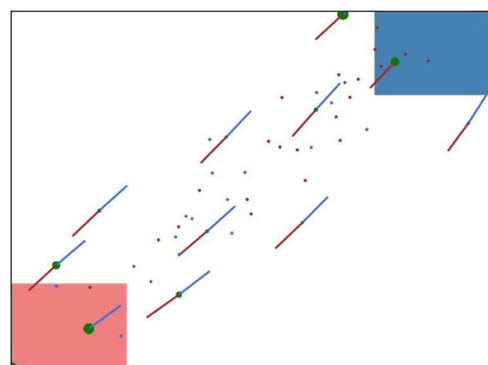
From Figs. 2 and 3 one can see how this controller increases the efficiency of the system. Here, the black lines indicate the paths traveled by 10 randomly selected foraging agents for the last 5 time steps. While the swarm is exploring, it uses ≈ 35 beacons before starting to re-configure. After around $100\,s$, beacons disappear from large unvisited areas

(a) Adapting swarm after 200s.

(b) Adapting swarm after 400s.

(c) Adapting swarm after 800s.

Fig. 3 Foraging in a dynamical environment

Table 4 Navigation delay measured for all agents for $[t_{hit}, T]$ for movable beacons

	Mean	Std
Static beacons	**12.51**	5.12
Movable beacons	**10.86**	3.47

Bold values indicate the mean of the results

well that the absolute optimal travel time between nest and food source would in this case be $\approx 9s$, and the re-configured swarm achieves 10.86

5.2.2 Non-stationary environments

The updating of information stored in the beacon grid forced by the evaporation factors, and the dynamical beacon grid generated by allowing beacons to move, should in theory allow our method to adapt to changing environments. Here, we present the experimental results regarding the adaptability of the system with respect to non-stationary environments. We considered a similar environment as in Ducatelle et al. (2011a) with two obstacles forcing a curved path between the targets. From Fig. 3 one can see that the system is able to create a sub-optimal trajectory with an efficient beacon infrastructure. After 201 seconds the obstacles are removed. From Fig. 3 one can conclude that 600 seconds after the environment changes, the system is able to create a new efficient beacon structure around the new optimal trajectory.

5.2.3 Agent failures and measurement errors

We now present the experiment results regarding the adaptability of the system with respect to agent failures and noisy measurements. First, to account for possible disturbances in sensor measurements, we consider the disturbed weight and velocity values for a random variable $\mu_k \sim \mathcal{N}(1, \sigma^2)$,

$$
\begin{aligned}
\tilde{\omega}_b^s(k, \sigma) &:= |\mu_k| \omega_b^s(k), \\
\tilde{\mathbf{v}}_f(k, \sigma) &:= D(\mu_k, \mu_k) \mathbf{v}_f(k).
\end{aligned}
\tag{17}
$$

That is, every measurement of weights transmitted by the beacons and every velocity measurement computed by the agents are perturbed proportionally with a random noise signal with variance σ^2. In this case we simulate the swarm without obstacles.

The results are presented in Table 5. One can see how as we increase the variance of the noise, the navigation delay worsens progressively. However, even for relatively "wide" normal distributions with $\sigma^2 = 1$, the swarm shows relative success at solving the foraging problem (recall that the absolutely optimal navigation delay is ≈ 8 s).

of the domain and the rest get closer to the optimal trajectory, allowing more accurate trajectories by the agents. When the system converges, the swarm is using ≈ 18 beacons, half of those in the static beacon scenario.

We compare in Table 4 the navigation delay for the same scenario with and without moving beacons. The distributed approach for adapting the beacon positions results in a significant reduction of navigation delay, indicating a more optimal trajectory set but specially a direct improvement in the total amount of trips for a similar size of the swarm, given that a number of beacons return to foraging functions. Reecall as

Table 5 Navigation delay for $[t_{hit}, T]$ for swarm with perturbed communication

σ^2	Mean	Std
0.01	**13.37**	2.62
0.1	**13.63**	3.04
0.5	**14.30**	1.02
1.0	**20.24**	0.76
2.5	**327.80**	67.29

Bold values indicate the mean of the results

Table 6 Navigation delay for $[t_{hit}, T]$ for swarm with temporal agent failure

n_{off} (%)	Mean	Std
1	**15.59**	7.44
5	**17.98**	10.58
10	**20.01**	12.94
20	**21.28**	11.31
50	**32.43**	6.36

Bold values indicate the mean of the results

Now we set to analyse a scenario where agents fail randomly in the swarm (both foragers and beacons). We implement this by considering the following failure scheme. Every $T_{off} = 20s$, a random fraction n_{off} of the agents "fails" during a total of T_{off} seconds. When an agent fails, it is unable to move, send or receive messages. After T_{off} seconds, the agents return to normal operation, and a new subset of agents is drawn at random and experiences failure. We repeat this for the entire $400s$ of the simulation, and measure the navigation delay of the swarm.

The results for the random agent failure scenario are presented in Table 6. The algorithm proposed presents significant robustness versus random agent failures. Observe that, even in the scenario where 50% of agents fail every $20s$, the performance of the swarm does not collapse, and in fact yields a navigation delay of only $2\times$ the baseline scenario. This would mean that, on average, the agents that are online solve the foraging problem almost with the same delay as the baseline scenario: the navigation delay is a measure of seconds per trip and agent, therefore having half the agents failing through the entire simulation would already yield a navigation delay at least twice as the baseline.

5.3 Robotic swarm

At last we implemented the work on real Elisa3-robots, in order to qualitatively confirm results of the work. Since the Elisa-3 robot is also used for the Webots experiments, the robot characteristics and behaviour are the same for both the real swarm and the simulated swarm. Note that the Elisa3-robots matches the restrictions of Assumption 1, except the capacity to measure its angular orientation. To overcome this

shortcoming, a global tracking system is employed which equips the robots with a virtual compass. Since our tracking system is only able to measure the orientations of the robots when all robots are not moving, the robots run synchronously.

In the test set-up, a swarm of 35 robots is deployed around a starting region and needs to look for a target in an arena of size 2.4 m $\times$ 1.15 m without obstacles (see Electronic Suplementary Material 1). Figure 4a and b shows snapshots of this experiment. The red and blue pillars are the centers of the nest region and target region, respectively. Green coloured robots are beacons, red coloured are searching for the nest region, blue coloured are searching for the target region. The black arrows indicate the heading direction, the blue and red arrows the guiding velocities $\mathbf{u}(k)$ at each beacon.

The resulting behaviour aligns with the predicted design dynamics. The swarm creates a covering beacon network. The first robots reaching the target region attracts other robots to this region and after 30 steps all non-beacon robots travel back and forth between the target regions, clustered around the shortest path. We point out that during all tests there were robot failures. This did not affected the swarm's behaviour noticeably, showing its robustness. We leave for future work the realisation of extensive tests with more powerful robots to confirm all results in other scenarios of swarm deployment.

6 Discussion

We have presented a foraging swarm design where robots are able to guide each-other to successfully forage without the need of position measurements, infrastructure or global knowledge, and using minimal amounts of knowledge about each-other or the environment. The system has been implemented on a swarm of Elisa-3 robots, and an extensive experimental study has been performed using the *Webots* simulator. We have shown how a middle sized swarm ($N \approx 100$) is able to find an unknown target and exploit trajectories close to the minimum length path. The system does require agents to know their orientation, and we have seen how it can be affected by overcrowding effects when agents need to avoid colliding with each-other. Additionally, we have observed how the optimality of the trajectories is affected by the resulting distribution of the beacons, which gives room for future work regarding the possibility of having beacons re-configure to more optimal positions. In Sect. 5.1 we show how the swarm is able to solve the foraging problem for an environment with and without obstacles, for different numbers of agents. Then, in Sect. 5.2 we demonstrate how the proposed system is able to adapt to dynamic environments, and the robustness it presents versus measurement noise and system faults. We expect in the future to add ultra-wide band communication modules to the Elisa-3 robots with magnetometers, that would allow us to run the system on a much
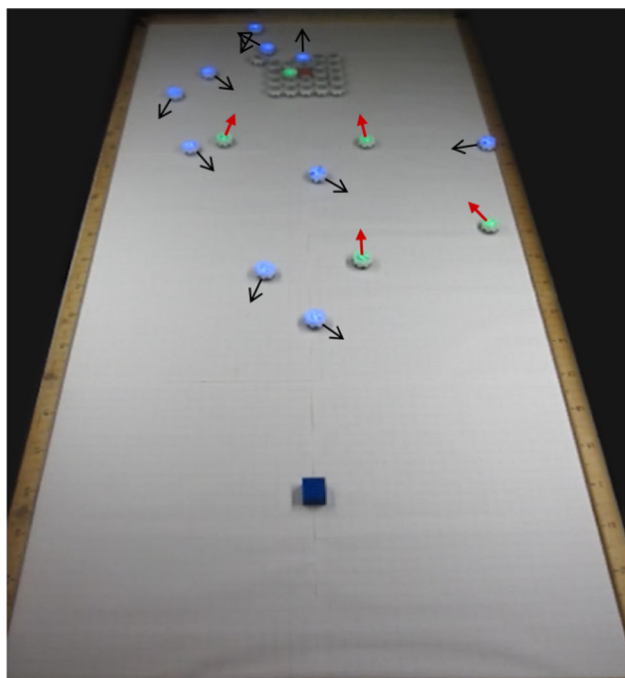
(a) Exploration
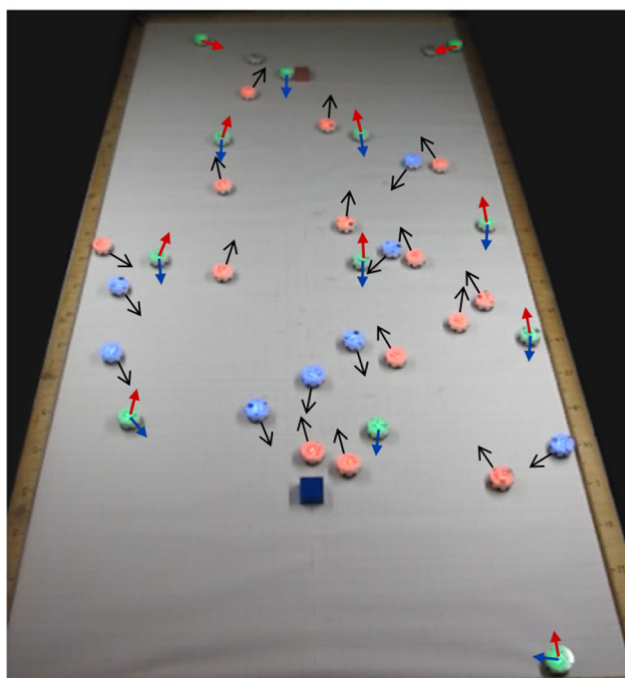
(b) Exploitation

Fig. 4 Robotic swarm with Elisa3 robots

larger swarm and on more complex environments, and to apply it to other navigation-based problems like surveillance, target interception or flocking. We leave for future work as well the formal analysis of the resulting beacon graphs, and the evolution of the variables $\omega(k)$ and $\mathbf{u}(k)$.

6.1 Advantages and shortcomings

We aim now to summarize the advantages this method presents with respect to existing literature, as well as short-comings that emerge from the design. First, we would like to emphasize the fact that the proposed system does not require line-of-sight communication, as done in e.g., Sperati et al. (2011); Hoff et al. (2013); Ducatelle et al. (2011a); Hrolenok et al. (2010); Lemmens and Tuyls (2009); Russell et al. (2015); Payton et al. (2001). This would enable the swarm to be deployed in much more diverse environments (cluttered, uneven terrain…), and we believe is behind the strong robustness results obtained in the experiments. Additionally, it does not require relative position measurements (Hrolenok et al., 2010; Payton et al., 2001; Ducatelle et al., 2011a; Lemmens and Tuyls, 2009; Hoff et al., 2013; Ducatelle et al., 2011b), which also adds significant flexibility when considering specific environments or robot types, and it does this while keeping foraging metrics in the line of the existing literature (in the best case, $\approx 1.5 \times -2\times$ the absolute optimal travel time). Additionally, we demonstrated experimentally how this simplicity of the swarm requirements results in a swarm that is significantly robust towards robot faults or dynamic environments. At last, the proposed structure for the foragers to become beacons (and beacons to go back to foragers) should lead to much lower amount of beacons required (in comparison to e.g.(Hrolenok et al., 2010; Hoff et al., 2013)), since we do not require the beacon network to be in range of each other, and we require an amount of *messages per step* of order $\mathcal{O}(N)$ that are asynchronous between robots.
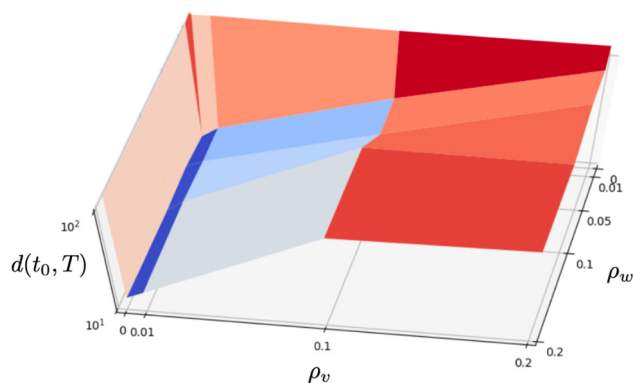
However, we do still retain the assumption of a global reference frame, that may not be able to be implemented with a compass in inside spaces. Additionally, the nature of the algorithm causes the robotic swarm's performance to decrease significantly when the environment becomes too cluttered with robots. At last, we still need to demonstrate the capacity of a real swarm of robots to operate in a large environment, since the Elisa3 robots proved difficult to be applicable to large swarms and arenas. Testing the algorithm in more complex robots (e.g., e-Puck robots) would probably result in much more relevant experiments.

Declarations

Conflict of interest   No other conflict of interest apply.

Fig. 5 Results for combinations of ρ values

A Parameter selection

The swarm design presented has a set of parameters that affect the behaviour and performance of the swarm. We set out now to address the choice of these parameters used for the simulations in this work. The final parameters used for all swarms are presented in Table 1.

A.1 Evaporation rate ρ

The evaporation rate ρ can be interpreted as the speed at which the system forgets about "past" information. It introduces a trade-off between how important "new" actions are versus how important old knowledge is. Therefore, we would want higher values of ρ when there is unknown information being discovered, and lower values to prevent de-stabilization of the swarm due to single agent errors. We fixed the rest of the parameters to the values of Table 1 for the considered environment with obstacle and run 50 particle simulation runs for different combinations of evaporation rates, computing the resulting navigation delay. We allowed for different evaporation rates in the weights (ρ_w) and the velocities (ρ_v), and the results are presented in Fig. 5. Therefore, a single value of $\rho = 0.01$ was chosen as relevant for the current experiments.

A.2 Diffusion rate λ

The diffusion rate influences how fast the information spreads through the network of beacons. It can be analogous to a discount rate in a reinforcement learning problem; larger values generate smoother gradients of the weight and velocity fields, and lower values generate sharper gradients. Intuitively, only a small subset of beacons receive a reward of r (they need to be close to $\mathcal{S}, \mathcal{T}$). Then, the foragers propagate this information through the network as they explore, and the information is *diffused* at a rate of $\lambda^{L(\mathcal{T},b)}$, where $L(\mathcal{T}, b)$ the minimum number of beacons that a forager needs to encounter from the food source $\mathcal{S}$ to beacon b. We are then interested on not losing too much information for large environments, therefore we can pick a λ close to 1 to ensure this, but strictly smaller than 1. In fact, in Ornia et al. (2022), authors show how for an ideal ant inspired swarm with number of agents $N \to \infty$, the influence of diffusion parameters vanishes as long as $0 > \lambda > 1$, and they show that for $\lambda > 1$ the weight values explode and grow unbounded as $k \to \infty$.

A.3 Reward r

The reward parameter mainly controls the scale of the weight values. Since the agents only reward those beacons in range of the target region with r, and the rest with 0, it serves as a scaling factor for the weight field. Very small values would create weight gradients through the network that are not strong enough to properly guide agents, where too large values could attract agents too fast towards wrong areas of the environment. A baseline value of $r = 1$ was tested and showed good results for the considered scenario. When having larger swarms and larger environments, it would be recommended to use larger r values to make sure the gradients are constructed on larger networks.

A.4 Exploration rate ε

The exploration rate controls the probability of choosing a random direction of movement for the foragers at every step k. Larger probabilities yield stronger exploration and worse foraging, but too low probabilities yield a risk of trapping agents in local minima or not solving the foraging problem correctly. A batch of simulations were performed to decide on the values of ϵ, and values between $\epsilon \in [0.03, 0.1]$ showed the best navigation delay for swarms between $N = 47$ and $N = 101$.

A.5 Sampling period τ

The sampling period affects both the communication rate and the rate of re-computation of velocities for the foraging robots. Intuitively, shorter periods are necessary for faster
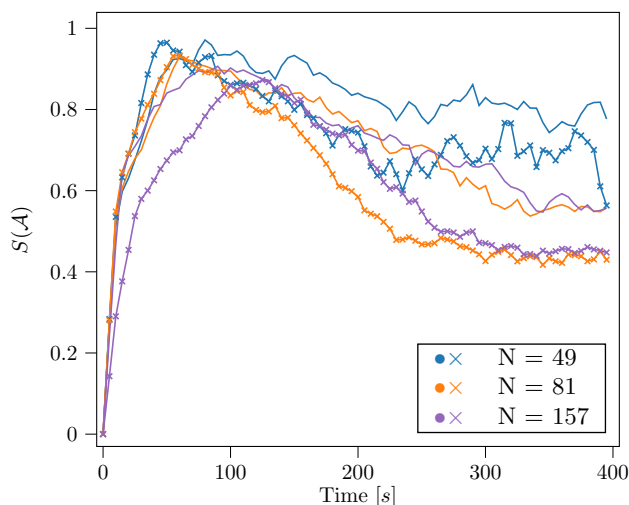
Fig. 6 Social entropy with and without ($\times$) obstacles

robots and smaller environments, and longer periods are suitable for slower robots or very large environments. A value of $\tau=1$ s was chosen to be representative of most system's capabilities (1 Hz communication frequencies) and reasonable for the velocities of the agents (they would communicate 4 times per meter of trajectory).

A.6 Instrument range δ

In practice, the instrument range is determined by the robotic platform used (However, this can be "virtually" limited if the robots are able to measure signal power). Larger values would require less beacons to cover the space, but the detail of the coverage would be much more coarse. We limited the range of signals to $\delta = 0.4m$ to consider a representative case where the robots can use other short range communication if necessary (e.g., Bluetooth), and to allow for a more general scenario where we need several robots to cover the entire environment.

B Entropy results

Accumulation of agents around trajectories (or path creation) is an example of emergent self-organized behaviour. To measure this, entropy has been used to quantize forms of clustering in robots, and to investigate if stable self-organization arises. We use the hierarchic social entropy as defined by Balch (2000) using single linkage clustering. Two robots are in the same cluster if the relative position between them is smaller than h. Then, with $\mathcal{C}(t, h)$ being the set of clusters at time t with minimum distance h, $c_i \in \mathcal{C}(t, h)$ and $\mathcal{A}_{c_i}$ being the subset of agents in cluster c_i, entropy for a group

of robots is

$$H(\mathcal{A}, h) = - \sum_{c_i \in \mathcal{C}(t,h)} \frac{|\mathcal{A}_{c_i}|}{N} \log_2 \left(\frac{|\mathcal{A}_{c_i}|}{N} \right). \quad (18)$$

Hierarchic social entropy is then defined as integrating $H(\mathcal{A}, h)$ over all values of h, $S(\mathcal{A}) = \int_{\delta_0}^{\infty} H(\mathcal{A}, h)dh$. To allow for zero entropy, we take the lower limit of the integral to be the diameter of the robots ($\delta_0 = 4$ cm). The entropy results of the experiments in Sect. 5.1 are presented in Fig. 6. In this case, we only compute the entropy of the forager agents, and we normalise against a practical "maximum" entropy computed by randomizing agents over the entire space. At $t = 0$ all agents start at the nest, hence the low entropy values, and from there the entropy increases while the swarm explores and tries to find the target. After the exploration phase, the entropy begins to settle to lower values as the robots accumulate over fewer trajectories. Entropy is higher for the symmetrical obstacle due to first, the split of agents among the two possible paths, and second, the minimum length path being longer.

C Extended concept

C.1 Beacon-forager switching

To introduce switching from the beacon state to a forager state, we first define the last foraging state of a beacon agent as $s_a^-(k)$. Then, the logic rules for state switching in (3) are updated as:

$$s_a(k + 1) =$$
$$\begin{cases} "B" & \text{if } \nexists b \in \mathcal{B}(k) : \mathbf{x}_b(k) \in \mathcal{D}_a(k) \wedge \\ & a \notin \mathcal{B}(k-2), \\ "F_1" & \text{if } s_a(k) = F_2 \wedge \mathbf{x}_a(k) \in \mathcal{S} \text{ or} \\ & a \in \mathcal{B}(k) \wedge \sum_s \omega_a^s(k) < \eta_\omega \wedge \\ & \nexists f \in \mathcal{A} : \mathbf{x}_f(k) \in \mathcal{D}_a(k) \wedge s^-(k) = F_1, \\ "F_2" & \text{if } s_a(k) = F_1 \wedge \mathbf{x}_a(k) \in \mathcal{T} \text{ or} \\ & a \in \mathcal{B}(k) \wedge \sum_s \omega_a^s(k) < \eta_\omega \wedge \\ & \nexists f \in \mathcal{A} : \mathbf{x}_f(k) \in \mathcal{D}_a(k) \wedge s^-(k) = F_2 \\ s_a(k) & \text{else} \end{cases} \quad (19)$$

with $\eta_w \in \mathbb{R}_+$. Compared to the old switching rules, two conditions for the foraging states are added, these work as follows: A beacon agent can only switch to a foraging state if there are no foraging agents within its region of influence and its summed stored weight values are smaller than some threshold value η_w.

After a beacon agent switches to a foraging state, it will always perform a random step, as by construction there is no other beacon within the agents range. Whether or not the agent will end up in the region of influence of another beacon

after the first step, is related to the number of former neighbouring beacons. If the former beacon was fully surrounded by beacons, the agent will always end up in the region of influence of a beacon after one step. Else, there is a chance of ending up in a region not covered by a beacon after the first step. To prevent an agent from switching directly back to the beacon state in such case, we restrict the switching from forager state to beacon state such that a forager agent needs to be for at least 2 time steps in a foraging state before it can switch back to the beacon state.

Last, we force an agent to 'forget' the weight and guiding velocity values after switching from the beacon state to a forager state, i.e. ω_a^s and $\mathbf{u}_a^s$ are set to zero after a state transition. Since the information stored at a beacon is location depended, preserving the former beacon information will only disturb the system if a former beacon agent transforms to a beacon again at another location.

D Experiment details

We use the Webots (Michel, 2004) simulator to perform the extensive statistical analysis presented in Sect. 5.1. The Webots simulator can simulate realistic robots in physical environments, including the Elisa-3 robot (GCTronic) (see Fig. 7) we used for the real implementation of the work.

The Elisa-3 robots are equipped with 8 IR proximity sensors (see Fig. 7) that are used for collision avoidance. Let us denote the event of detection of an obstacle by sensor # as *Trigger_Prox_#*. The collision avoidance algorithm used for both the Webots simulations and the robotic implementation of the work is then given by Algorithm 3. The algorithm ensures that the maximum distance traveled per time step is never greater than τv_0, i.e., $\|\mathbf{x} - \mathbf{x}_1\|_2 \leq \tau v_0$. Consequently, the formal guarantees presented in Sect. 4 still hold for a swarm that performs collision avoidance.
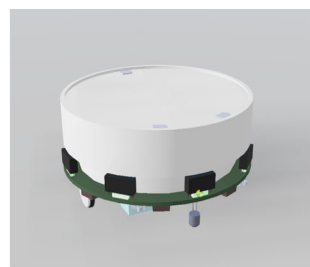

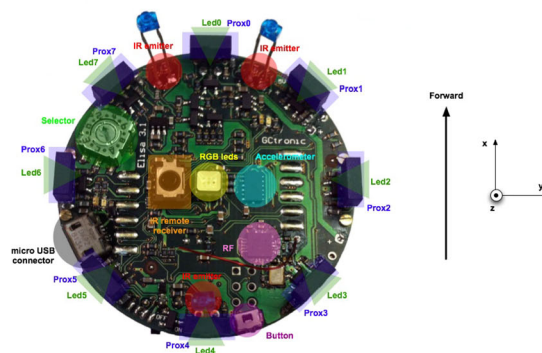
Algorithm 3 Collision avoidance Elisa-3 robot

---

if Trigger_Prox_1 **then**
    Turn 10° left
else if Trigger_Prox_7 **then**
    Turn 10° right
else if Trigger_Prox_0 **then**
    Turn 180°
else if Trigger_Prox_5 **then**
    Turn 10° left
else if Trigger_Prox_3 **then**
    Turn 10° right
else if Trigger_Prox_4 **then**
    Turn 180°
end if

(a) Webots model of Elisa-3 Robot.

(b) Hardware components of an Elisa-3 Robot. The IR proximity sensors are highlighted by blue boxes and tagged as 'Prox#'.

Fig. 7 Elisa-3 robot

References

Alers, S., Tuyls, K., Ranjbar-Sahraei, B., et al. (2014). Insect-inspired robot coordination: foraging and coverage. *Artificial life, 14*, 761–768.

Balch, T. (2000). Hierarchic social entropy: An information theoretic measure of robot group diversity. *Autonomous robots, 8*(3), 209–238.

Beni, G., Wang, J. (1993). Swarm intelligence in cellular robotic systems. In *Robots and biological systems: Towards a new bionics?* Springer, pp. 703–712

Blum, C., Merkle, D. (2008). Swarm intelligence. In Blum, C., Merkle, D. (Eds.) *Swarm Intelligence in Optimization*, pp 43–85

Campo, A., Gutiérrez, Á., Nouyan, S., et al. (2010). Artificial pheromone for path selection by a foraging swarm of robots. *Biological cybernetics, 103*(5), 339–352.

Dorigo, M., Birattari, M., et al. (2007). Scholarpedia. *Swarm intelligence, 2*(9), 1462.

Drogoul, A., Ferber. J. (1993). Some experiments with foraging robots. In From Animals to Animats 2: Proceedings of the Second International Conference on Simulation of Adaptive Behavior, MIT Press, p. 451

Ducatelle, F., Di Caro, GA., Pinciroli, C., et al. (2011b). Communication assisted navigation in robotic swarms: Self-organization and cooperation. In 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, pp. 4981–4988

Ducatelle, F., Di Caro, G. A., Pinciroli, C., et al. (2011). Self-organized cooperation between robotic swarms. *Swarm Intelligence, 5*(2), 73–96.

Ducatelle, F., Förster, A., Di Caro, G. A., et al. (2009). Supporting navigation in multi-robot systems through delay tolerant network communication. *IFAC Proceedings Volumes, 42*(22), 25–30.

Font Llenas, A., Talamali, MS., Xu, X., et al. (2018). Quality-sensitive foraging by a robot swarm through virtual pheromone trails. In *Swarm Intelligence*. Springer International Publishing, pp. 135–149

Fujisawa, R., Imamura, H., Hashimoto, T., et al. (2008). Communication using pheromone field for multiple robots. In 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, pp. 1391–1396

Fujisawa, R., Dobata, S., Sugawara, K., et al. (2014). Designing pheromone communication in swarm robotics: Group foraging behavior mediated by chemical substance. *Swarm Intelligence, 8*(3), 227–246. https://doi.org/10.1007/s11721-014-0097-z

Garnier, S., Tache, F., Combe, M., et al. (2007). Alice in pheromone land: An experimental setup for the study of ant-like robots. In 2007 IEEE Swarm Intelligence Symposium, pp. 37–44, https://doi.org/10.1109/SIS.2007.368024

Hoff, N., Wood, R., Nagpal, R. (2013). Distributed colony-level algorithm switching for robot swarm foraging. In Distributed autonomous robotic systems. Springer, pp. 417–430

Hrolenok, B., Luke, S., Sullivan, K., et al. (2010). Collaborative foraging using beacons. In AAMAS, pp. 1197–1204

Ishii, S., Yoshida, W., & Yoshimoto, J. (2002). Control of exploitation-exploration meta-parameter in reinforcement learning. *Neural networks, 15*(4–6), 665–687.

Jarne Ornia, D., Mazo, M. (2020). Convergence of ant colony multi-agent swarms. In Proceedings of the 23rd International Conference on Hybrid Systems: Computation and Control. Association for Computing Machinery, New York, NY, USA, HSCC '20, https://doi.org/10.1145/3365365.3382199

Johansson, R., Saffiotti, A. (2009). Navigating by stigmergy: A realization on an RFID floor for minimalistic robots. In 2009 IEEE International Conference on Robotics and Automation, IEEE, pp. 245–252

Kennedy, J. (2006). Swarm intelligence. In *Handbook of nature-inspired and innovative computing*. Springer, pp. 187–219

Kuffner, J. J., LaValle, S. M. (2000). Rrt-connect: An efficient approach to single-query path planning. In Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065), IEEE, pp. 995–1001

Lemmens, N., de Jong, S., Tuyls, K., et al. (2007). Bee system with inhibition pheromones. In European conference on complex systems, Citeseer

Lemmens, N., Tuyls, K. (2009). Stigmergic landmark foraging. In Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems-Vol. 1, pp. 497–504

Mayet, R., Roberz, J., Schmickl, T., et al. (2010). Antbots: A feasible visual emulation of pheromone trails for swarm robots. In M. Dorigo, M. Birattari, G. A. Di Caro, et al. (Eds.), *Swarm Intelligence* (pp. 84–94). Springer.

Michel, O. (2004). Cyberbotics ltd. webots™: professional mobile robot simulation. *International Journal of Advanced Robotic Systems, 1*(1), 5

Nair, A., McGrew, B., Andrychowicz, M., et al. (2018). Overcoming exploration in reinforcement learning with demonstrations. In 2018 IEEE International Conference on Robotics and Automation (ICRA), IEEE, pp. 6292–6299

Nouyan, S., Campo, A., & Dorigo, M. (2008). Path formation in a robot swarm. *Swarm Intelligence, 2*(1), 1–23.

Ornia, DJ., Zufiria, PJ., Mazo Jr., M. (2022). Mean field behavior of collaborative multiagent foragers. *IEEE Transactions on Robotics*

Payton, D., Daily, M., Estowski, R., et al. (2001). Pheromone robotics. *Autonomous Robots, 11*(3), 319–324.

Reina, A., Cope, A. J., Nikolaidis, E., et al. (2017). ARK: Augmented reality for Kilobots. *IEEE Robotics and Automation Letters, 2*(3), 1755–1761.

Russell, RA. (1997). Heat trails as short-lived navigational markers for mobile robots. In Proceedings of International Conference on Robotics and Automation, IEEE, pp. 3534–3539

Russell, K., Schader, M., Andrea, K., et al. (2015). Swarm robot foraging with wireless sensor motes. In Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, Citeseer, pp. 287–295

Sperati, V., Trianni, V., & Nolfi, S. (2011). Self-organised path formation in a swarm of robots. *Swarm Intelligence, 5*(2), 97–119.

Sugawara, K., Kazama, T., Watanabe, T. (2004). Foraging behavior of interacting robots with virtual pheromone. In 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566), IEEE, pp. 3074–3079

Svennebring, J., & Koenig, S. (2004). Building terrain-covering ant robots: A feasibility study. *Autonomous Robots, 16*(3), 313–332.

Talamali, M. S., Bose, T., Haire, M., et al. (2020). Sophisticated collective foraging with minimalist agents: A swarm robotics test. *Swarm Intelligence, 14*(1), 25–56.

Thrun, S. B. (1992). *Efficient exploration in reinforcement learning*. Technical Report, USA

Ziparo, VA., Kleiner, A., Nebel, B., et al. (2007). RFID-based exploration for large robot teams. In Proceedings 2007 IEEE International Conference on Robotics and Automation, pp. 4606–4613, https://doi.org/10.1109/ROBOT.2007.364189

Steven Adams is a Ph.D. Candidate at the Delft Center for Systems and Control, Delft University of Technology (The Netherlands). He received a BSc in Mechanical Engineering at Delft University of Technology in 2016, a MSc in Econometrics and Operations Research from the Vrije Universiteit Amsterdam in 2020 and a MSc in Systems and Control from Delft University of Technology in 2021. His main research interests are multi-agent learning, multi-robot systems and bayesian deep learning models.
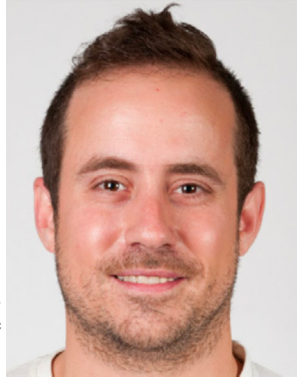
Daniel Jarne Ornia is a Ph.D. Candidate at the Delft Center for Systems and Control, Delft University of Technology (The Netherlands). He received a BSc in Aerospace Engineering at Universitat Politecnica de Catalunya in Barcelona in 2015, and his MSc in Aerospace Engineering from the Royal Institute of Technology (KTH) in Stockholm in 2017. Prior to joining TU Delft as a PhD student he worked as a data scientist on AI industry applications. His main research interests are learning multi-agent systems, and the study of efficient communication and information sharing in collaborative systems.

Manuel Mazo is an associate professor at the Delft Center for Systems and Control, Delft University of Technology (The Netherlands). He received the Ph.D. and M.Sc. degrees in Electrical Engineering from the University of California, Los Angeles, in 2010 and 2007 respectively. He also holds a Telecommunications Engineering "Ingeniero" degree from the Polytechnic University of Madrid (Spain), and a "Civilingenjor" degree in Electrical Engineering from the Royal Institute of Technology (Sweden), both awarded in 2003. Between 2010 and 2012 he held a joint post-doctoral position at the University of Groningen and the innovation centre INCAS3 (The Netherlands). His main research interest is the formal study of problems emerging in modern control system implementations, and in particular the study of networked control systems and the application of formal verification and synthesis techniques to control. He has been the recipient of a University of Newcastle Research Fellowship (2005), the Spanish Ministry of Education/UCLA Fellowship (2005–2009), the Henry Samueli Scholarship from the UCLA School of Engineering and Applied Sciences (2007/2008) and an ERCStarting Grant (2017).