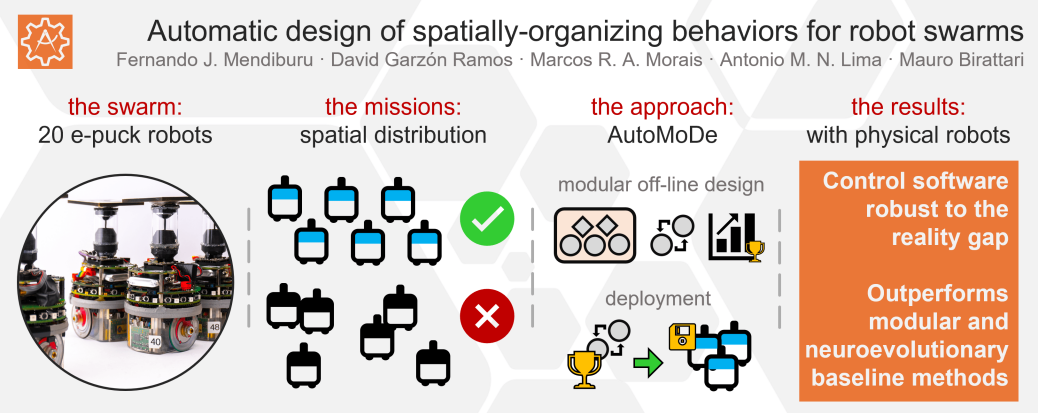
Cite as: Fernando J. Mendiburu, David Garzón Ramos, Marcos R. A. Morais, Antonio M. N. Lima, and Mauro Birattari (2022). AutoMoDe-Mate: automatic off-line design of spatially-organizing behaviors for robot swarms. *Swarm and Evolutionary Computation*, 74, 101118. DOI: 10.1016/j.swevo.2022.101118

Graphical Abstract

AutoMoDe-Mate: automatic off-line design of spatially-organizing behaviors for robot swarms

Fernando J. Mendiburu, David Garzón Ramos, Marcos R. A. Morais, Antonio M. N. Lima, Mauro Birattari



Email addresses: fernando.mendiburu@ee.ufcg.edu.br (Fernando J. Mendiburu), dgarzonr@ulb.ac.be (David Garzón Ramos), morais@dee.ufcg.edu.br (Marcos R. A. Morais), amnlima@dee.ufcg.edu.br (Antonio M. N. Lima), mbiro@ulb.ac.be (Mauro Birattari)

These authors contributed equally: Fernando J. Mendiburu and David Garzón Ramos.

Corresponding authors: Fernando J. Mendiburu, David Garzón Ramos, and Mauro Birattari.

Highlights

AutoMoDe-Mate: automatic off-line design of spatially-organizing behaviors for robot swarms

Fernando J. Mendiburu, David Garzón Ramos, Marcos R. A. Morais, Antonio M. N. Lima, Mauro Birattari

- Automatic methods can design spatially-organizing behaviors for robot swarms.
- AutoMoDe-Mate is a design method specialized in spatial organization problems.
- AutoMoDe-Mate includes a module that enables robot pattern formation.
- Experiments with physical robots show effectiveness in coverage and networking.
- AutoMoDe-Mate outperforms baseline modular and neuro-evolutionary methods.

Email addresses: fernando.mendiburu@ee.ufcg.edu.br (Fernando J. Mendiburu), dgarzonr@ulb.ac.be (David Garzón Ramos), morais@dee.ufcg.edu.br (Marcos R. A. Morais), amnlima@dee.ufcg.edu.br (Antonio M. N. Lima), mbiro@ulb.ac.be (Mauro Birattari)

These authors contributed equally: Fernando J. Mendiburu and David Garzón Ramos.

Corresponding authors: Fernando J. Mendiburu, David Garzón Ramos, and Mauro Birattari.

Cite as: Fernando J. Mendiburu, David Garzón Ramos, Marcos R. A. Morais, Antonio M. N. Lima, and Mauro Birattari (2022). AutoMoDe-Mate: automatic off-line design of spatially-organizing behaviors for robot swarms. *Swarm and Evolutionary Computation*, 74, 101118. DOI: 10.1016/j.swevo.2022.101118

AutoMoDe-Mate: automatic off-line design of spatially-organizing behaviors for robot swarms

Fernando J. Mendiburu[a,b,1,2], David Garzón Ramos[a,1,2], Marcos R. A. Morais[b], Antonio M. N. Lima[b], Mauro Birattari[a,2]

[a] *IRIDIA, Université Libre de Bruxelles, Av. Franklin Roosevelt 50, Brussels, 1050, Belgium*
[b] *Universidade Federal de Campina Grande, R. Aprígio Veloso, 882, Campina Grande, 58428-830, Paraíba, Brazil*

Abstract

We present `Mate`, an automatic off-line design method specialized in the design of spatially-organizing behaviors for robot swarms. `Mate` belongs to the family of modular methods known as AutoMoDe. We introduce `Mate` to study the automatic design of collective behaviors for missions in which the swarm is subject to spatial distribution constrains. In this paper, we produce control software for three missions with specifications related to the distribution of the swarm in the environment. We conduct experiments in simulation and with a swarm of 20 e-puck robots. Alongside `Mate`, we also conduct experiments with two other automatic design methods: `Chocolate`—a state-of-the-art instance of AutoMoDe; and `EvoSpace`—a method based on neuroevolution. Early studies conducted with existing modular design methods have shown their limitations in the design of spatially-organizing behaviors for robots that operate under spatial constrains. By introducing a specialized method like `Mate`, we expect to overcome these limitations. The aggregate results of our experiments show that `Mate` performs significantly better than `Chocolate` and `EvoSpace` in the missions we consider.

Email addresses: `fernando.mendiburu@ee.ufcg.edu.br` (Fernando J. Mendiburu), `dgarzonr@ulb.ac.be` (David Garzón Ramos), `morais@dee.ufcg.edu.br` (Marcos R. A. Morais), `amnlima@dee.ufcg.edu.br` (Antonio M. N. Lima), `mbiro@ulb.ac.be` (Mauro Birattari)

[1] These authors contributed equally: Fernando J. Mendiburu and David Garzón Ramos.
[2] Corresponding authors.

July 20, 2022

Keywords: swarm robotics, automatic design, spatially-organizing behaviors, relative positioning, AutoMoDe, evolutionary robotics

1. Introduction

A robot swarm [1, 2] is a self-organizing system that can adapt to the environment and exhibit a wide variety of collective behaviors [3]. We investigate to which extent automatic off-line design is a viable approach to designing robot swarms that operate under spatial distribution constrains. To this end, we introduce AutoMoDe-`Mate` (`Mate` hereafter): an automatic modular method specialized in the design of spatially-organizing behaviors.

Swarm robotics is an approach to the decentralized coordination of large groups of robots [2]. In a robot swarm, the collective behavior emerges from the interactions that individual robots have with their peers and with their environment [4]. Robot swarms have desirable properties such as redundancy, flexibility, fault-tolerance, parallelism, and scalability [2, 3]. Unfortunately, no general methodology exists to define what an individual robot must do so that a specific collective behavior emerges [3]. Therefore, the control software of robot swarms is primarily handcrafted and the intuition and experience of the designer play a major role in the design process—which remains costly and time consuming. Automated engineering methodologies are crucial to overcome this limitation [5].

Current perspectives on the future of swarm robotics emphasize the need to leverage the realization of robot swarms with automatic methods [5–7]. In a recent work, we also discussed how automatic methods are a viable approach to designing collective behaviors for robot swarms [8]. In comparison to other research lines in swarm robotics, little work is devoted to developing automated engineering methodologies to conduct/assist the design process [6]. Automatic methods generate collective behaviors via optimization—that is, an optimization algorithm evaluates possible control software for the robots and selects an appropriate instance to perform a desired mission. Traditionally, research in automatic design has focused on the neuro-evolutionary approach [9, 10]. In this approach, the control software of the robots has the form of an artificial neural network whose parameters (and possibly architecture) are obtained via evolutionary computation. More recently, a few modular design methods have been introduced as an alternative to neuro-evolution [11–21]. In the modular approach, the control software results

from the tuning of pre-defined software modules and their combination into specific architectures—e.g., probabilistic finite-state machines [11–19, 21, 22] or behavior trees [20, 21]. Neuro-evolution and modular design are both appealing approaches to the automatic generation of robot control software. A larger body of literature belongs to the former [6] and, only recently, studies started to shed light on how popular methods of the two approaches compare to each other [23].

Spatially-organizing behaviors concern the organization and distribution of robots and objects in space [3]. Robots displaying spatially-organizing behaviors are characterized by their ability to establish accurate relative positioning with respect to their peers [24]. Typical examples are behaviors such as aggregation [11, 25], chain-formation [26, 27], self-assembly and morphogenesis [28, 29], and object clustering and assembling [30, 31]. In particular, we are interested in cases where the swarm displays a regular and repetitive spatial distribution of robots [32, 33]—often refereed to as pattern-formation [3]. The generation of spatially-organizing behaviors was originally tested in the first studies of automatic modular design [12]. However, results were not satisfactory when the robots had to operate under constrains in their relative positioning. Our hypothesis are that existing modular methods either (i) do not have a proper set of modules to address this class of missions or, if they have them, the methods fail at (ii) properly combining them in good-performing control software. No automatic modular method has been conceived to specifically address the design of spatially-organizing behaviors.

Automatic modular design is a general framework that must be specialized to address a pre-defined class of problems with a specific robot platform [34]. Studies in automatic modular design have shown that the approach can effectively produce specialized collective behaviors—for example, in missions related to collective exploration [14], communications [15], and the interaction of robots with objects in their environment [17]. Commonly, researchers establish a class of problems (i.e., a class of missions) of interest and conceive an appropriate modular design method to address it. To conceive a modular design method, one must define three principal components: (i) the robot platform for which control software should be designed; (ii) the optimization algorithm that drives the design process; and (iii) the control architecture and software modules that can be combined to produce the control software of the robots. We expect that by introducing a new software module that facilitates the relative positioning of robots, `Mate` will

become a design method specialized in the design of spatially-organizing behaviors for robot swarms. In this sense, we expect that `Mate` will overcome the limitations observed in the previously proposed methods [12].

`Mate` belongs to the family of modular design methods known as AutoMoDe [11]. Like other instances of this family [12–15, 17], `Mate` produces control software by assembling pre-defined software modules into probabilistic finite-state machines. We developed `Mate` on the basis of `Chocolate` [12]—a state-of-the-art instance of AutoMoDe. In addition to the original set of modules conceived for `Chocolate`, `Mate` includes a new module that enables the hexagonal pattern-formation of robots. We develop the new module via a principled design method [24] based on virtual physics [35]—more precisely, the Lennard-Jones potential [36]. This new module allows `Mate` to address missions related to the spatial distribution of the robots.

Alongside `Mate`, we also present results obtained with other two methods for the automatic design of robot swarms: `Chocolate`, as implemented by Francesca *et al.* [12]; and `EvoSpace`, an original neuro-evolutionary method that—likewise `Mate`—incorporates the Lennard-Jones potential. `EvoSpace` is an implementation of the neuro-evolutionary approach that builds on `EvoStick`—a method that has been used as a baseline in many other studies [11, 12, 18, 20, 22]. We conduct our study following the recently proposed tenets of the automatic off-line design of robot swarms [8]: (i) the design methods we consider can address a whole class of missions without undergoing any modification; and, (ii) once a mission is specified, no human intervention is provided for in any phase of the design process. In the absence of a well established state of the art in the automatic design of spatially-organizing behaviors, we find that `Chocolate` and `EvoSpace` are appropriate baselines. We assess `Mate`, `Chocolate` and `EvoSpace` in three missions where the performance of the swarm depends on the ability of the robots to operate under spatial distribution constrains. We present the results of experiments with a swarm of 20 e-puck robots, both in simulation and reality. The rest of the paper is organized as follows: we discuss related work in Section 2; in Section 3, we introduce `Mate`; we describe the experimental setup in Section 4; in Section 5, we present the results and discussion; and in Section 6, we highlight the conclusions and future work.

2. Related Work

Spatially-organizing behaviors have been largely studied in the context of multi-agent systems [37], multi-robot systems [38, 39], and swarm robotics [3]. The literature in these domains is extensive and therefore we restrict our attention to studies in swarm robotics [3, 4]. In particular, we focus on studies that are relevant to the automatic design of robot swarms [8, 40].

We conceive `Mate` to design collective behaviors that resemble those observed in previous studies on pattern formation [41–43]. Pattern-formation behaviors are commonly conceived under the framework of virtual physics—in particular, with methods that use artificial potential fields [35]. In this approach, robots react to attractive and/or repulsive virtual forces. The artificial potential field approach was proposed by Khatib [44], and then adapted to swarm robotics by Spears and Gordon [35]. Artificial potential fields have been used in the context of monitoring and surveillance [45, 46], distributed sensing and actuation [47], coverage [46, 48], and collective motion [45, 49–51] among others. Designers commonly produced these collective behaviors following an ad hoc procedure. That is, designers manually produced control software and evaluated it only on a specific mission. As a consequence, results are difficult to reproduce and it is unclear whether they are of general applicability to a wider range of problems. Manual design prevents the generalization and use of collective behaviors beyond the missions for which they are originally conceived [8, 40, 52]. Alternatively, automatic design can provide generally applicable methodologies to design collective behaviors [3].

`Mate` is an automatic off-line design method—a particular case of the optimization-base design of robot swarms [40]. In optimization-based design, the control software of the robots is produced via an optimization process. Common classifications divide optimization-base design into (i) on-line and off-line methods and into (ii) semi-automatic and (fully) automatic methods. On-line methods produce control software while the robots operate in the target environment and off-line methods produce the control software prior to the deployment of the robots—typically by using simulations. In semi-automatic methods, a human designer uses an optimization algorithm as their primary design tool and, conversely, automatic methods do not require human intervention during the design process. These classifications are not meant to be considered as strict—indeed, hybrids exists. These classifications are rather a convenient way to set appropriate performance expectations and to make fair comparisons across methods.

The literature in the optimization-based design of spatially-organizing behaviors belongs mostly to the fields of evolutionary robotics and neuro-evolution [9, 10]. For example, Duarte *et al.* [33, 53–55] produced hybrid and hierarchical control software by evolving individual behaviors that are then executed via a behavior arbitrator. In a latter study, the authors used this method to design spatially-organizing behaviors for robot swarms that detect intruders in their workspace—both in simulation [54] and with physical robots [33, 55]. There are also studies that consider the design of spatially-organizing behaviors with virtual physics and artificial potential fields [56–60]—mainly in the case of semi-automatic design. For example, Pinciroli *et al.* [59] produced control software for swarms of satellites that self-organize in lattices, and latter, the authors used artificial evolution to fine-tune the parameters of the control software [60]. Unfortunately, although neuro-evolution is a versatile approach, it is prone to suffer from bootstrapping and deception [53]. Also, it has been observed that the performance of neuro-evolutionary methods is substantially affected by the so-called *reality gap* [11, 22, 23, 53, 61, 62]—that is, the difference between simulations models and reality.

Automatic modular design methods are less affected by the reality gap than implementations of the neuro-evolutionary approach [12, 15, 17, 18, 20, 23]. In previous work, the modular approach has been used to design collective behaviors in various classes of missions. For example, `Chocolate` has being tested in the design of aggregation [11, 16, 18, 20–22], foraging [11, 16, 18–22], decision making [15], collective exploration [14], and coverage [12] behaviors. With respect to spatially-organizing behaviors, Francesca *et al.* [12] tested `Chocolate` in three missions where robots must operate under spatial distribution constrains: SURFACE AND PERIMETER COVERAGE; COVERAGE WITH FORBIDDEN AREAS; and LARGEST COVERING NETWORK. In SURFACE AND PERIMETER COVERAGE, the swarm performs well if the robots uniformly cover the perimeter and surface of two target regions. Similarly, in COVERAGE WITH FORBIDDEN AREAS, the swarm performs well if the robots uniformly cover the space that is not indicated as forbidden. And finally, in LARGEST COVERING NETWORK, the performance of the swarm is proportional to the surface covered by the largest network of robots—i.e., robots that maintain connectivity at a specific distance. Although `Chocolate` can design control software for this class of missions, results are not completely satisfactory. In most cases, the robots address the mission without achieving any meaningful spatial organization. `Chocolate`—like other instances of

AutoMoDe—has limitations in the design of spatially-organizing behaviors with relative positioning between robots. With `Mate`, we expect to overcome this limitation.

3. AutoMoDe-`Mate`

`Mate` designs control software in the form of probabilistic finite-state machines: the states are low-level behaviors that a robot can perform; and the transitions are conditions to switch from one behavior to another. Both low-level behaviors and transitions are pre-defined software modules: they are defined once and for all in a mission-agnostic way. Starting from the specification of a mission, `Mate` searchers for an appropriate combination of these modules and returns an instance of control software that is then uploaded to the robots. In the following, we first characterize the robot platform for which `Mate` designs control software; then, we introduce the pre-defined software modules on which `Mate` is based; and finally, we describe the automatic design process.

3.1. Robot platform

`Mate` produces control software for an extended version of the e-puck [63], a two-wheeled differential-drive robot designed for research and education. We consider a version of the e-puck that is equipped with a Linux extension board [64], a range-and-bearing board [65], an omnidirectional camera [66], RGB LEDs, three infrared ground sensors, and eight infrared proximity sensors that also measure ambient light intensity [63]. Methods in the AutoMoDe family operate over an abstraction of the robotic platform that we refer to as a *reference model* [67]. The reference model formalizes the inputs and outputs of the control software, and establishes its relationship with the sensors and actuators of the robot. Although `Mate` builds on the top of `Chocolate`, `Mate` designs control software for the reference model RM 3.1 and `Chocolate` does it for RM 1.1 [67]. The two methods consider e-pucks with (formally) different sensing and actuation capabilities and therefore they operate with different reference models. In RM 3.1, e-pucks use their omnidirectional camera and RGB LEDs to estimate the relative position of other e-pucks. Table 1 describes RM 3.1 and highlights the differences with respect to RM 1.1.

In RM 3.1, the velocity of the wheels ($v_{i \in \{left,right\}}$) of the robot can be set independently in the range $[-0.12,0.12]$ m/s. The 3 ground sensor mea-

Table 1: RM 3.1. Novelties with respect to RM 1.1 are highlighted in gray. The possible values of r_{prox}, r_{light}, $r_{R\&B}$, r_{cam} are given in the form (range, bearing).

Input	Value	Description
r_{prox}	$([0,1], \angle[0,2]\,\pi\,\text{rad})$	proximity direction vector
r_{light}	$([0,1], \angle[0,2]\,\pi\,\text{rad})$	light direction vector
$r_{R\&B}$	$([0,1], \angle[0,2]\,\pi\,\text{rad})$	neighboring robot direction vector
n	[0,20]	number of neighboring robots perceived, $n \in \{1,..,20\}$
gnd	$\{black, gray, white\}$	reading of the ground sensor
r_{cam}	$([0,1], \angle[0,2]\,\pi\,\text{rad})$	camera direction vector

Output	Value	Description
v_i	[−0.12,0.12]	target linear wheel velocity (m/s), with $i \in \{left, right\}$
led	$\{on, off\}$	LEDs emitting cyan light

Period of the control cycle: 0.1 s.

surements (*gnd*) indicate whether the robot is on black, gray, or white floor. The robot can detect nearby objects using its 8 proximity sensors; their readings are aggregated by the vector $r_{prox} = \sum_{i=1}^{8}(|r_{prox_i}|, \angle r_{prox_i})$, if objects are detected; and $r_{prox} = (1, \angle 0)$, otherwise. The 8 light sensors indicate the direction of an ambient light source, if present. These sensors are sensitive to infrared light and do not perceive the LEDs of other robots. The readings of the light sensors are aggregated in the vector $r_{light} = \sum_{i=1}^{8}(|r_{light_i}|, \angle r_{light_i})$, if light is detected; and $r_{prox} = (1, \angle 0)$, otherwise. The vectors r_{prox} and r_{light} are defined in the range $([0, 1], \angle[0, 2]\,\pi\,\text{rad})$. The range-and-bearing board allows the robot to locate neighboring peers. The readings of the range-and-bearing board are aggregated in the vector $r_{R\&B} = \sum_{k=1}^{n}(|r_{R\&B_k}|, \angle r_{R\&B_k})$, if neighbors are detected; and $r_{R\&B} = (1, \angle 0)$, otherwise. In this case, $|r_{R\&B_k}| = (1 + r_k)^{-1}$, where $r_k \in [0, 0.4]\,\text{m}$ is the distance between a robot and the k^{th} neighboring robot (with $k \in \{1, ..., 20\}$) perceived by its range-and-bearing board. The robot can also estimate the number of neighboring robots in a range of 0.4 m. The LEDs of the robot (*led*) can be turned *on* or *off*. When turned *on*, they emit cyan light. The omnidirectional camera can perceive the LEDs of neighboring robots and can estimate their relative position. The vector r_{cam} aggregates the readings of the camera in a unique direction vector that is computed by using the Lennard-Jones force law—details of this computation are given later in Section 3.2, Equation 2.

3.2. Pre-defined software modules

`Mate` includes the six low-level behaviors and six transitions originally conceived for `Vanilla` and then adopted in `Chocolate`—for an in-depth description, see Francesca *et al.* [11, 12]. In addition to these modules, we include in `Mate` a new low-level behavior named FORMATION. The module FORMATION enables pattern formation. First, we briefly describe all modules included in `Mate` and then we present FORMATION in detail.

Behaviors:

- EXPLORATION: the robot randomly explores the environment;
- STOP: the robot stands still;
- PHOTOTAXIS: the robot goes towards the ambient light source, if perceived;

- ANTI-PHOTOTAXIS: the robot goes away from the ambient light source, if perceived;

- ATTRACTION: the robot goes towards its neighboring peers, if perceived;

- REPULSION: the robot goes away from its neighboring peers, if perceived;

- FORMATION: the robot maintains a fixed distance from its neighboring peers, if perceived.

EXPLORATION, PHOTOTAXIS, ANTI-PHOTOTAXIS, ATTRACTION, and REPULSION, and FORMATION embed an obstacle avoidance sub-behavior that prevents the robots to collide with their peers or with objects in their environment.

Transitions:

- BLACK-FLOOR: transition triggered if the floor is black;

- GRAY-FLOOR: transition triggered if the floor is gray;

- WHITE-FLOOR: transition triggered if the floor is white;

- NEIGHBOR-COUNT: transition triggered if sufficiently many neighboring robots are perceived;

- INVERTED-NEIGHBOR-COUNT: transition triggered if sufficiently few neighboring robots are perceived;

- FIXED-PROBABILITY: transition triggered with a fixed probability.

3.2.1. FORMATION: a module for robot pattern formation

FORMATION is a behavior in which the robot is subject to virtual forces that are computed through the Lennard-Jones potential [36]. As in other implementations of the artificial potential field approach [45], a group of robots executing FORMATION tends to adopt a spatial configuration that minimizes its overall potential energy. That is, robots aim to remain in equilibrium positions that are reached when the swarm is homogeneously distributed in space. A robot executing FORMATION reacts to virtual forces that originate from one or more neighboring peers executing FORMATION as

well. FORMATION aggregates all the virtual forces acting on the robot and computes a desirable displacement vector.

The Lennard-Jones force law (f_{ik}) can be used to conceive a highly parametrizable behavior model [51]—see Equation 1.

$$f_{ik} = -\frac{2\alpha\epsilon}{d_{ik}}\left[\left(\frac{d^*_{ik}}{d_{ik}}\right)^{2\alpha} - \left(\frac{d^*_{ik}}{d_{ik}}\right)^{\alpha}\right]. \tag{1}$$

It allows the definition of an arbitrary *target distance* (d^*_{ik}) between the robot i and any k^{th} neighboring peer, the intensity (ϵ) of the forces acting on the robot, and the steepness (α) of the transition between attraction and repulsion forces. If the distance between two robots is larger than the target distance, the robots will be subject to an attractive force. Conversely, if the distance is shorter, the robots will be subject to a repulsive force. When the robots are separated by the target distance, they remain in equilibrium and no force acts on them. Indeed, the model enables the individual parametrization of the attraction and repulsion forces [51, 68] and facilitates a smooth transitions between them [69].

In our implementation, e-pucks that execute FORMATION display the color cyan with their LED's. An e-puck can perceive and locate neighboring peers executing FORMATION with its omnidirectional camera. The vector r_{cam} (Equation 2) is computed through an average sum of the Lennard-Jones force on the set of vectors $r_{cam_k} = (|r_{cam_k}|, \angle r_{cam_k})$, for the all robots k perceived with the camera.

$$r_{cam} = \begin{cases} -\frac{1}{n}\sum\limits_{k=1}^{n}\left(\frac{2\alpha\epsilon}{d_k}\left[\left(\frac{d^*}{d_k}\right)^{2\alpha} - \left(\frac{d^*}{d_k}\right)^{\alpha}\right], \angle r_{cam_k}\right), & n \geq 1; \\ (1, \angle 0), & n = 0. \end{cases} \tag{2}$$

The parameter d is the distance measured by the camera, $d_k \in [0.05, 0.30]$ m, $\angle r_{cam_k} \in [0, 2]\,\pi$ rad, for each robot k in the set of robots $k \in \{1, ..., n\}$ perceived by the camera. We set $\alpha = 2$ and $\epsilon = 2.5$ according to values reported in the literature [51]. Robots executing FORMATION aim to remain equidistant at a target distance (d^*) with respect to neighboring peers. We expect that by doing so, they will tend to form hexagonal patterns—as it has been previously reported [70]. The size and density of the patterns depend on the target distance specified by d^*. Large values of d^* form large and sparse lattices of robots, and conversely, small values form small and dense

ones. In our experiments, the target distance (d^*) between robots is tuned by the automatic design process in the range $d^* \in [0.07, 0.25]$ m.

FORMATION sums the effects of the Lennard-Jones force law and a short-range obstacle avoidance sub-behavior that is also embedded in the module. The desirable displacement vector (r) is the sum of a displacement vector computed through the Lennard-Jones force law (r_{cam}) and a repulsion vector computed through the proximity sensors of the robot (r_{prox})—see Equation 3. The vector (r) is afterward translated into velocity commands according to RM 3.1:

$$r = r_{cam} - 5r_{prox}. \tag{3}$$

We consider the Lennard-Jones force law as an appropriate model to build a new software module for `Mate`: it requires a minimal set of sensors and actuators to be implemented [45]; and it only relies on local interactions [35]. We expect that the automatic design process will select the module and fine-tune the target distance between robots to better address missions with spatial distribution constrains.

3.3. Design of control software

`Mate` designs control software using Iterated F-race [71]—the optimization algorithm originally used in `Chocolate`. Other AutoMoDe methods have been conceived with alternative optimization algorithms—for example, simulated annealing [16] and iterative improvement [21]; yet, Iterated F-race remains a *de facto* standard optimization algorithm in the AutoMoDe family [34]. Iterated F-race explores and evaluates possible combinations of the software modules according to a mission-specific performance metric—a measure of the degree of success of the swarm in the mission at hand. When the design process starts, Iterated F-race samples the design space for candidate control software. More precisely, Iterated F-race samples candidate finite-state machines that result from the various combinations of the 15 software modules available in `Mate` (7 behaviors and 6 transitions) and the possible instantiation of their parameters. `Mate`, likewise `Chocolate`, can assemble finite-state machines of up to four behaviors—each of which with four outgoing transitions at most. During the optimization process, Iterated F-race evaluates each finite-state machine over a set of independent simulation runs and ranks the candidate solutions according to a series of Friedman tests [72]. Iterated F-race iteratively fine-tunes high-performing candidate

solutions and discards those solutions that have low performance and that are consistently outranked by others. The design process terminates when Iterated F-race has exhausted a maximum number of simulation runs. After that, `Mate` returns the best instance of control software found so far—which is then ported to the physical robots without undergoing any modification. For a more detailed discussion on the optimization process in the automatic modular design of robot swarms, we refer the reader to Francesca *et al.* [11], Kuckling *et al.* [16, 21], and Birattari *et al.* [34].

4. Experimental setup

We conduct our research both in simulation and with physical e-pucks. In this section, we describe the missions we consider and the experimental protocol we follow to assess `Mate`.

4.1. Missions

We conceived three missions to assess `Mate`: ANY-POINT CLOSENESS, NETWORKED COVERAGE, and CONDITIONAL COVERAGE. ANY-POINT CLOSENESS and NETWORKED COVERAGE are variants of missions already proposed by Francesca *et al.* [12], and CONDITIONAL COVERAGE is a mission that we propose here for the first time and that has been inspired by a collective decision making mission proposed by Hasselmann *et al.* [15]. The three missions present challenges that are similar to those that one would face in real-world coverage missions [46, 48] like monitoring and surveillance [45, 46] and distributed sensing and actuation [47]. As detailed later in this section, the missions we selected impose spatial constrains to the operation of the robot swarm and differ in complexity with respect to each other.

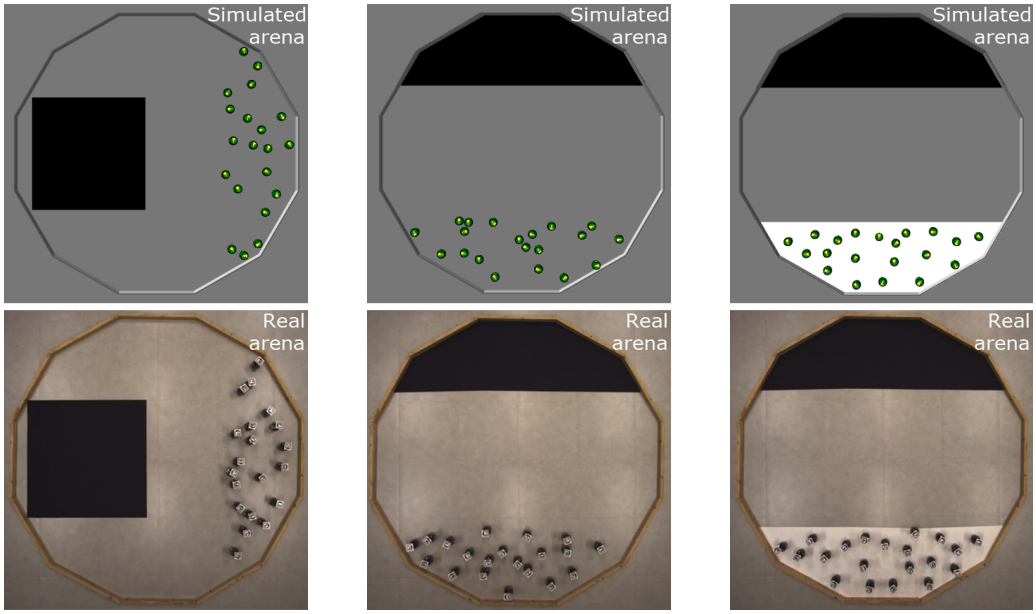
We focused on abstracting challenges related to how the robots would be required to selectively cover and maintain presence in certain areas of their operating environment. In ANY-POINT CLOSENESS, the robots must engage in uniformly covering a specific designated region in their operating environment during a limited time frame. This mission resembles those possible in monitoring and surveillance, in which the swarm should be capable of (i) maintaining a continuous and uniform presence in the designated area and of (ii) devoting all resources (i.e., robots) available to it. NETWORKED COVERAGE grows in complexity with respect to ANY-POINT CLOSENESS. In NETWORKED COVERAGE, in addition to cover a designated region, the robot swarm must establish a network of robots that maintain an arbitrary distance

between themselves. This challenge is particularly relevant to the deployment and operation of sensor networks: a robot swarm should maximize the sensing (covered) area while maintaining the robots in a sufficiently close distance so that the information can be continuously transmitted between them. Finally, we believe that CONDITIONAL COVERAGE is the more complex of the three missions. In CONDITIONAL COVERAGE, the swarm is presented with two designated regions from which only one must be covered. The complexity of this mission lies in the fact that the region to be covered is determined by the initial conditions of the mission—which differ from one execution to the other. In this sense, the robots should be capable of operating in the two cases, and collectively select the appropriate behavior each time. As one could expect in a real-world deployment, a robot swarm should be capable to adopt different behaviors without the need of being reprogrammed. For example, this could be the case of monitoring missions in which the swarm can autonomously adopt different coverage strategies that fit the characteristics of the region to be covered (e.g., size, terrain).

We expected that `Mate` would design collective behaviors that allow the swarm to achieve a meaningful spatial organization and effectively perform the missions. In our experiments, we produce control software for a swarm of 20 e-pucks—a swarm size consistently used in automatic design studies [11, 12, 14, 15, 17, 18, 20]. In most AutoMoDe methods demonstrated so far, the number of robots is a parameter that is given to the design process—although it is possible to tune it alongside the control software of the robots [13]. In `Mate`, the number of robots is part of the specifications of the mission and it is not fine-tuned by the optimization process, although an extension in this sense—inspired by [13]—would be readily possible. In the three missions, the e-pucks operate in a dodecagonal arena of about $4.9\,\mathrm{m}^2$ surrounded by walls. The floor is gray, and might contain black and white regions. The presence of black and/or white regions is specified on a per-mission basis and they denote the *target region* where robots must operate while exhibiting spatially-organizing behaviors. All missions are executed in a time $T = 120\,\mathrm{s}$. Fig. 1 shows the configuration of the arena in the three missions.

4.1.1. ANY-POINT CLOSENESS

In this mission, the robot swarm must uniformly cover a *target region* in the arena. The target region is a black square area of $1\,\mathrm{m}^2$. At the beginning of each experiment, the robots are randomly positioned in the right side of the arena—see Fig. 1a.



(a) ANY-POINT CLOSENESS (b) NETWORKED COVERAGE (c) CONDITIONAL COVERAGE

Figure 1: Simulated and real arena for the missions: ANY-POINT CLOSENESS, NETWORKED COVERAGE, and CONDITIONAL COVERAGE. The images show an example of the initial position of the robots. In (a,b), the black area indicates the target region where the robots must operate. In (c), the black and white areas are both target regions, but only one is designated at a time. See Section 4.1 for details.

The best performance is achieved when the following objective function is minimized

$$f_1(T) = E[d(T)]^2, \tag{4}$$

where $f_1(T)$ is the square of the *expected distance* $E[d(T)]$, measured only once at the end of the experiment. $E[d(T)]$ is the expected distance, at time T, between a generic point within the target region and its closest robot inside the target region [12]. We estimate/approximate $E[d(T)]$ by computing the average distance between *any point* within the target region (in practice, a sufficiently large set of randomly sampled points) and its closest robot inside the target region. More precisely, $E[d(T)] \approx \sum_{p=1}^{P} min_i(d_{ip})/P$; where $p \in [1, 1000]$ is a point in a set of points randomly and uniformly sampled from within the target region; and $i \in [1, 20]$ is a robot in the set of robots that are inside of the target region at time T. The distance d_{ip} is, therefore, the distance between a random point p and a robot i that are both within the target region.

4.1.2. NETWORKED COVERAGE

In this mission, the swarm must establish a network of robots that covers a target region in the arena. The *target region* is a black area of about $1\,\mathrm{m}^2$. The robot swarm must maximize, at every time, the ratio between the area covered by the robots in the network and the total area of the target region. The coverage range for each robot is smaller than the distance at which the robot can establish a network. We consider that each robot covers a circular area of 0.15 m radius. When robots are in the target region, they establish a network if they are at 0.3 m from each other. The area covered by the swarm is the sum of the area covered by all the robots. At the beginning of each experiment, robots are randomly positioned in the bottom side of the arena—see Fig. 1b.

The best performance is achieved when the following objective function is maximized

$$f_2(t) = \sum_{t=0}^{T} A_{N_B}(t)/A_{TR}, \tag{5}$$

where $f_2(t)$ is the sum at every time step of the ratio between the area $A_{N_B}(t)$ of the target region that is covered by the network of robots N_B, and the total area of the target region ($A_{TR} = 1\,\mathrm{m}^2$). N_B is the network of robots with

the largest number of individuals within the target region. The performance is measured at every time step ($\Delta t = 0.1\,\mathrm{s}$).

4.1.3. CONDITIONAL COVERAGE

In this mission, the robot swarm must selectively cover one out of two target regions in the arena. The target regions are a black and a white area of about $1\,\mathrm{m}^2$—see Fig. 1c. The target region to cover is conditioned at each experimental run by the starting position of the robots: if the robots start the experiment in the black area, they must travel to the opposite side of the arena and cover the white one; conversely, if robots start the experiment in the white area, they must travel to the opposite side and cover the black one. The robot swarm must maximize, at every time step, the ratio between the area of the target region they cover and its total area. Each robot covers a circular area of 0.15 m radius.

The best performance is achieved when the following objective function is maximized

$$f_3(t) = \sum_{t=0}^{T} A_m(t)/A_{TR}, \tag{6}$$

where $f_3(t)$ is the sum at every time step of the ratio between the area $A_m(t)$, covered by the m robots within the target region, and the total area of the target region ($A_{TR} = 1\,\mathrm{m}^2$). The performance is measured at every time step ($\Delta t = 0.1\,\mathrm{s}$).

4.2. Baseline methods: `Chocolate` *and* `EvoSpace`

Alongside `Mate`, we include in our study two baseline methods for the automatic design of robot swarms: `Chocolate` and `EvoSpace`. The baseline methods are the starting point from which we can draw some conclusions about the relative performance of `Mate` with respect to the existing literature. We expect that `Mate` will perform better than the baselines, as it is a method specialized in the class of missions that are subject in our experiments.

4.2.1. `Chocolate`

`Chocolate` is the automatic design method that we extended to create `Mate`, and both belong to the family of methods AutoMoDe. Originally presented by Francesca *et al.* [12], `Chocolate` has been used as a baseline to compare other instances of AutoMoDe [14–16, 20]. `Mate` and `Chocolate` differ in the robot platform for which they design control software and

in the set of modules they include. `Chocolate` does not include the low-level behavior FORMATION and does not have access to the omni-directional camera and LEDs of the robots. The rest of software modules and robot hardware remains the same for the two methods.

In principle, `Chocolate` has the low-level behaviors that are required enable the robots for maintaining a constant relative distance: ATTRACTION and REPULSION. The Lennard-Jones model we embed in FORMATION enables the robots to maintain a specific distance between each other by setting a particular combination of attraction and repulsion forces. `Chocolate` could generate finite-state machines that combine and instantiate ATTRACTION and REPULSION in a similar way. To produce such behavior, we expected that `Chocolate` would produce a fine-tuned combination of software modules so that the robots maintain the required constant relative distance between each other—for example, by constantly transitioning between ATTRACTION and REPULSION. However, we also expected that `Chocolate` (being a non-specialized method) would not perform as good as `Mate` and the Lennard-Jones model.

In our research, we use the implementation of `Chocolate` described by Francesca *et al.* [12]. `Chocolate` operates unmodified on the basis of the RM 1.1 of the e-puck, as defined earlier by the same authors [73].

4.2.2. `EvoSpace`

As shown in Section 2, many studies in the optimization-based design of robot swarms belong to the neuro-evolutionary approach. For this reason, we wished to investigate how a rather reasonable implementation of this approach would perform on the set of missions on which we assess `Mate`. We conceived `EvoSpace`: a neuro-evolutionary method for the automatic design of robot swarms that integrates the virtual physics model of `Mate`. Likewise `Mate` extends `Chocolate`, `EvoSpace` extends `EvoStick` —a neuro-evolutionary method previously used as a baseline to study `Chocolate` [12]. The artificial neural network in `EvoSpace`, in contrast to `EvoStick`, includes input information obtained from the Lennard-Jones model also embedded in `Mate`'s FORMATION; see Section 3.2. More precisely, it includes as an input the projections of the direction vector r_{cam} defined in Equation 2. We expected that `EvoSpace` could, in principle, design control software that benefits from this input information to produce similar spatially-organizing behaviors than those that are expected from `Mate`. Analogous specializations of `EvoStick` have been used to investigate the automatic design of

communication behaviors for e-pucks that can communicate using the range-and-bearing [15] and their omni-directional camera and LEDs [17].

`EvoSpace` is a method conceived to produce control software for the reference model RM 3.1 of the e-puck. Likewise `EvoStick`, `EvoSpace` produces control software in the form of a fully-connected feed-forward artificial neural network. The neural network considered in `EvoSpace` integrates the Lennard-Jones model defined in Equation 2. It has 15 input nodes, 4 outputs nodes, and no hidden layers. The input nodes correspond to 4 inputs for a projection vector (r_{prox}) of the proximity readings, 4 inputs for a projection vector (r_{light}) of the ambient light readings, 4 inputs for a projection vector (r_{cam}) of the camera readings, 1 input for the number of robots perceived by the range-and-bearing (n), 1 input for the readings of the ground sensor (gnd), and 1 bias input node. In all cases, the projection vector results from projecting the sensor readings in four unit vectors that point at 45°, 135°, 225°, and 315° with respect to the robot coordinate system. The output nodes correspond to 2 outputs that control the velocity of the wheels ($v_{i\in\{left,right\}}$), and 2 outputs that control the state of the LEDs (led). `EvoSpace` tunes the synaptic weights of the neural network via artificial evolution—using the evolutionary algorithm implemented for `EvoStick` [73]. Likewise `Mate`, `EvoSpace` optimizes the neural network until a pre-defined budged of simulation runs is exhausted.

Although more popular and advanced neuro-evolutionary methods exist (e.g., `CMA-ES` [74], `xNES` [75], and `NEAT` [76]), it has been observed that their advanced features do not provide any practical advantage over `EvoStick` [23]—in both, the single-layer and multi-layer perception cases. For this reason, we deemed `EvoStick` an appropriate starting algorithm to build `EvoSpace`. Because of the results published in [23], we do not have any reason to expect that a more complex implementation of `EvoSpace` would yield higher performance in the automatic design of robot swarms.

4.3. Protocol

We compare `Mate`, `Chocolate`, and `EvoSpace`. For each mission, we execute each design method 10 times to obtain 10 instances of control software. All methods have a budget of 200 000 simulation runs to produce each instance of control software. After obtaining all the instances of control software, we assess their performance once in simulation and once with physical robots. Simulations are performed using ARGoS3, version 48 [77], together

with the argos3-epuck library [78]. ARGoS3 is a fast, parallel, and multi-engine simulator specifically designed for multi-robot systems and swarm robotics [79]. In the experiments with physical robots, we use a tracking system [80] to track the robots in the experimental arena. Both in simulation and in the experiments with the physical robots, we compute the performance of the robot swarm on each experimental run using ARGoS3. In simulation runs, ARGoS3 estimates the position of the simulated robots at every time step, then it processes the information according to the objective function specified in each mission (see Section 4.1), and afterward it returns the performance value. In experimental runs with physical robots, the tracking system [80] provides ARGoS3 with the position of the physical robots at every time step. Then, likewise the simulations, ARGoS3 processes the information and returns the performance value. The computation made by ARGoS to estimate the performance of the swarm does not vary between the simulation and the experiments with physical robots.

We present numerical results with box-and-whiskers boxplots that show the experimental results on a per-mission basis. For each method, we report the performance obtained in simulation and with physical robots—thin and thick boxes, respectively. We support the mission-specific performance comparison of methods with Wilcoxon paired rank sum tests [72] at a 95% of confidence. In addition, we present a Friedman test that aggregates the overall performance of the methods across the three missions.

We also describe the spatial organization that the robots display in the experiments—analysis conducted by visual inspection. In the context of this paper, references to *a stable state* imply that the swarm remains in a state in which we do not appreciate major disturbances in the relative positioning of the robots. Conducting a particular stability analysis is not in the focus of our study and is beyond the scope of this paper. We support our discussion with the performance statistics defined above and with the videos we provide as supplementary material [81].

We also provide additional information about the final spatial distribution of the robots on a per-mission basis. More precisely, we investigate if the robots are prone to remain outside the target region and, if not, in which ratio they cross the perimeter of the region. The perimeter of the target region is the area of about 0.15 m (twice the diameter of the robot) that is closer to the gray floor of the arena. We use pictures to illustrate the final spatial distribution of the robots—the full set of pictures is provided as supplementary material [81]. In the pictures, we overlay the target region (lined

in green) and the perimeter (lined in yellow) of the arena. For each mission, we also estimate the percentage of the robots of the swarm that remain outside the target region at the end of each experimental run. We present the results with cumulative frequency plots for the three design methods. Considering the robots that reach the target region, we estimate the percentage of those robots that cross the perimeter. We also present these results with cumulative frequency plots.

Finally, the analysis of the control software produced with `Chocolate` and `Mate` is also supported with heat-map plots obtained in simulation (Fig. 8). The plots show the average percentage of usage of all behavior modules in the three missions. To produce the heat-map plots, we aggregate the time that robots spend in all the behavior modules across the 10 experimental runs. We display the results in windows of 100 time steps.

5. Results

In this section, we present the results of our experiments. First, we discuss quantitatively and qualitatively the results for each mission and method. Then, we elaborate on the aggregate results across the three missions.

5.1. ANY-POINT CLOSENESS

Fig. 2 (I) shows the numerical results of ANY-POINT CLOSENESS. The control software produced by `Mate` performs significantly better than the one produced by `Chocolate`—both in simulation and reality. If `Mate` is compared with `EvoSpace`, there is no statistical evidence to conclude that `Mate` performs better than `EvoSpace` when assessed in simulation. However, `Mate` performs significantly better than `EvoSpace` in the experiments with physical robots.

Fig. 2 (II) shows the cumulative frequency of the percentage of robots that remained outside the target region at the end of each experimental run. Fig. 2 (III) shows the cumulative frequency of the percentage of robots in the target region that crossed the perimeter. The cumulative frequency plots show that, among the three methods, `Chocolate` is less prone to leave robots outside the target region at the end of the experiment, followed by `Mate`, and then by `EvoStick`. `Mate` and `EvoSpace` appear to design collective behaviors in which a similar percentage of the robots in the target region cross the perimeter. The percentage of robots in the target region that cross the perimeter tends to be higher for `Mate` and `EvoSpace` than for `Chocolate`.

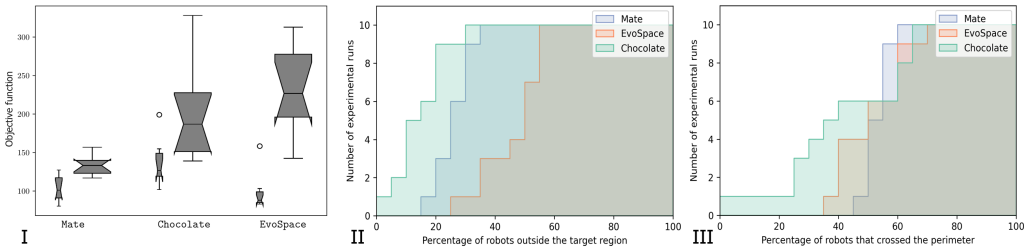


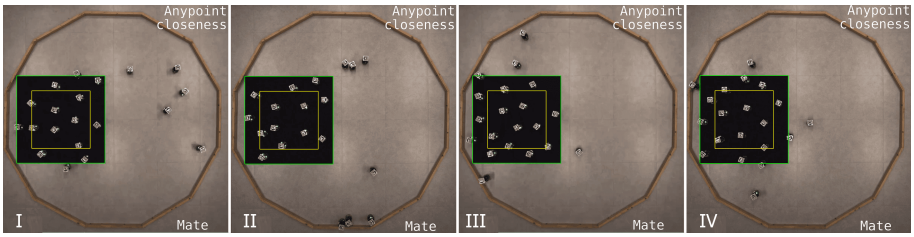
Figure 2: ANY-POINT CLOSENESS. (I) Box-and-whiskers plots of the results obtained with `Mate`, `Chocolate` and `EvoSpace`. Thin boxes represent results obtained in simulation and thick boxes those obtained with physical robots. The lower, the better. (II) Cumulative frequency of the percentage of robots that remained outside the target region at the end of each experimental run. (III) Cumulative frequency of the percentage of robots in the target region that crossed the perimeter.

Fig. 3 shows illustrative examples of the spatial distribution of the robots at the end of four experimental runs of ANY-POINT CLOSENESS.

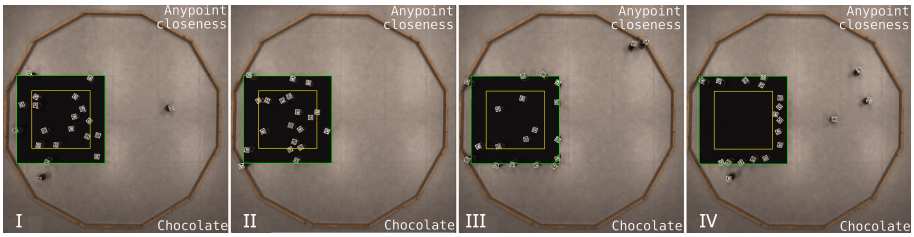
`Mate` designs collective behaviors in which the robots cover the target region while maintaining a specific relative distance from their peers. The robots explore the arena until they step into the target region. Then, they position themselves considering nearby robots that also entered the region. The swarm uniformly covers the space by steadily using FORMATION; see Fig. 3a (I–IV). Once the target region is highly dense, further robots remain outside of it and do not disrupt the spatial distribution of their peers.

`Chocolate` mainly designs individualistic behaviors in which the robots move into the target region without considering the position of their peers. The robots explore the arena and when they step in the target region, they stop after a certain period of time and remain still in place until the end of the experiment (Fig. 3b). As the robots do not coordinate with their peers, the swarm does not cover the target region uniformly; see Fig. 3b (I,II). In some cases, the robots mostly remain in the perimeter of the target region and form a barrier; see Fig. 3b (III,IV).

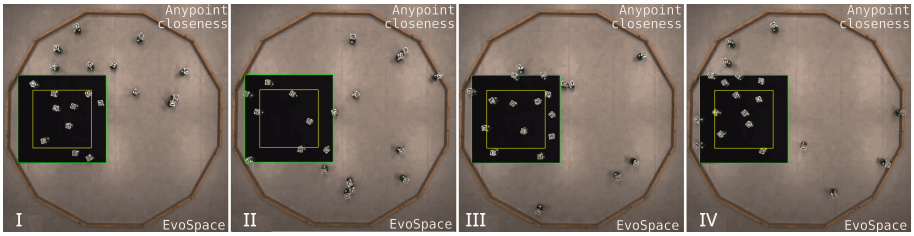
In the case of `EvoSpace`, the robots also explore the arena until they step into the target region. Once a robot enters the region, it switches to an in-place rotation behavior—we understand this behavior as a mean to stand still in a fixed location. Contrary to the behavior observed in `Chocolate`, in `EvoSpace` the robots remain reactive after stopping. Robots that reach the target region react to the presence of incoming robots that push them further. However, we do not observe a stable and uniform spatial distribution of



(a) `Mate`



(b) `Chocolate`



(c) `EvoSpace`

Figure 3: Illustrative examples of the final spatial distribution of robots in ANY-POINT CLOSENESS. Each row shows results obtained with four different instances of control software produced by (a) `Mate`, (b) `Chocolate`, and (c) `EvoSpace`. The target region is lined in green and the perimeter is lined in yellow.

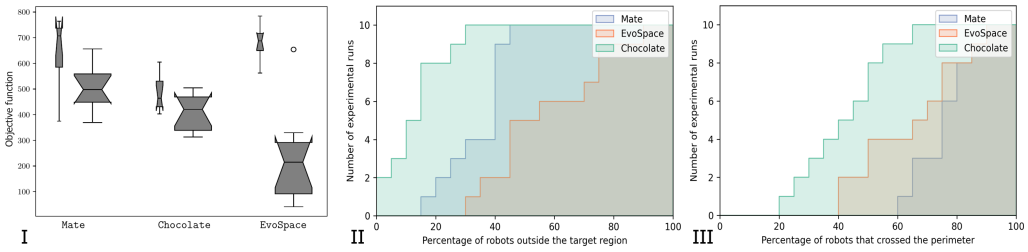


Figure 4: NETWORKED COVERAGE. (I) Box-and-whiskers plots of the results obtained with `Mate`, `Chocolate` and `EvoSpace`. Thin boxes represent results obtained in simulation and thick boxes represent results obtained with physical robots. The higher, the better. (II) Cumulative frequency of the percentage of robots that remained outside the target region at the end of each experimental run. (III) Cumulative frequency of the percentage of robots in the target region that crossed the perimeter.

robots; see Fig. 3c (III,IV). `EvoSpace` designed collective behaviors in which the robots are not very efficient in exploring the arena. That is, a large number of robots often do not reach the target region; see Fig. 3c (I,II).

As we expected, the performance of the swarm is highly conditioned by the ability of the robots to position themselves with respect to their peers. In this mission, the distribution of robots in the target region is more relevant than the number of robots inside it. `Mate` designs collective behaviors that distribute the robots more uniformly than those designed by `Chocolate` and `EvoSpace`. The capability of `Mate` to design such behaviors reflected in a better performance. Despite that `Chocolate` designed collective behaviors in which more robots reach the target region, the swarm could not distribute uniformly. Considering the results obtained in simulation and with physical robots, the control software produced by `Mate` crosses the reality gap better than the one produced by `Chocolate` and `EvoSpace`.

5.2. NETWORKED COVERAGE

Fig. 4 (I) shows the numerical results of NETWORKED COVERAGE. The control software produced by `Mate` performs significantly better than the one produced by `Chocolate`—both in simulation and reality. When assessed in simulation, there is no significant difference between the control software produced by `Mate` and `EvoSpace`. On the other hand, when the design methods are assessed with physical robots, `Mate` performs significantly better than `EvoSpace`.

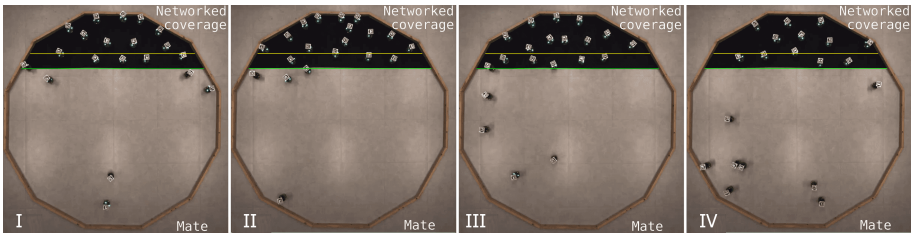
Fig. 4 (II) shows the cumulative frequency of the percentage of robots

that remained outside the target region at the end of each experimental run. Fig. 4 (III) shows the cumulative frequency of the percentage of robots in the target region that crossed the perimeter. When comparing the three methods, the cumulative frequency plots show that `Chocolate` is less prone to leave robots outside the target region at the end of the experiment—followed by `Mate`, and then by `EvoStick`. A similar result to that observed in ANYPOINT CLOSENESS. However, in this mission, the percentage of robots in the target region that cross the perimeter tends to be higher for `Mate`, followed by `EvoSpace` and then by `Chocolate`. Fig. 5 shows illustrative examples of the spatial distribution of the robots at the end of four experimental runs of NETWORKED COVERAGE.

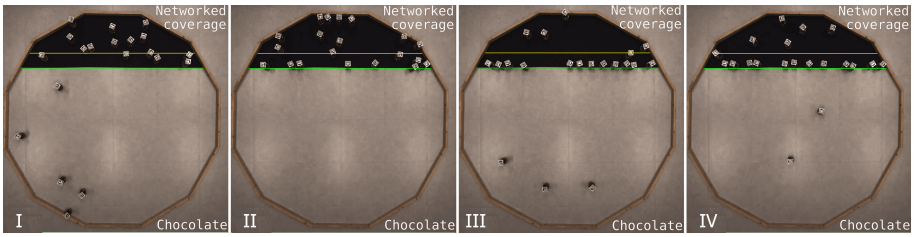
`Mate` designs collective behaviors in which the robots remain at a fixed distance between each other; see Fig. 5a (I–IV). By doing so, the swarm establishes networks of robots that maintain connectivity in a range of 0.3 m—as required in the specifications of the mission. The robots establish/join a network by executing FORMATION once they enter the target region. The network remains stable as the robot density in the target region increases, and robots outside progressively join the network without affecting the overall distribution of the swarm.

`Chocolate` designs collective behaviors in which the robots may adopt two roles: (i) robots that randomly walk and locate within the target region—see Fig. 5b (I,II); and (ii) robots that remain in a standstill behavior on the edge of it—see Fig. 5b (III-IV). The combination of robots adopting the two roles allows the swarm to place together a large number individuals. The barrier of robots that line up along the border of the target region encloses the robots that move inside of it, and cause them to establish a network. After a visual inspection, we argue that this solution has two drawbacks: first, robots have difficulties to enter the target region and join the network once the barrier of robots is settled; and second, the networks are not stable because robots inside the target region continuously move and do not maintain the connectivity. As a result, the swarm fails to establish a network of robots that efficiently covers the entire target region.

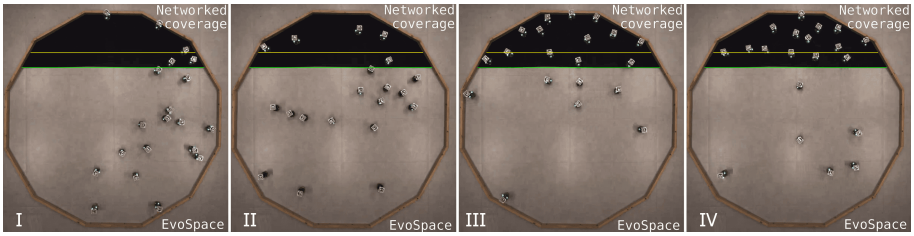
`EvoSpace` mainly designs collective behaviors in which the robots explore the arena until they step into the target region, and afterwards, they remain in place by continuously rotating. Most of the control software produced by `EvoSpace` relies on fine-tuned exploration behaviors that let the robots navigate the arena—*e.g.*, wall following behaviors. When this control software is ported to the physical robots, it strongly suffers the effects of the reality gap



(a) `Mate`



(b) `Chocolate`



(c) `EvoSpace`

Figure 5: Illustrative examples of the final spatial distribution of robots in NETWORKED COVERAGE. Each row shows results obtained with four different instances of control software produced by (a) `Mate`, (b) `Chocolate`, and (c) `EvoSpace`. The target region is lined in green and the perimeter is lined in yellow.

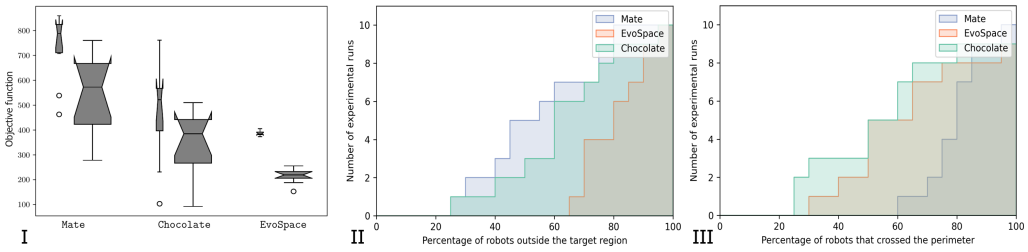


Figure 6: CONDITIONAL COVERAGE. (I) Box-and-whiskers plots of the results obtained with `Mate`, `Chocolate` and `EvoSpace`. Thin boxes represent results obtained in simulation and thick boxes represent results obtained with physical robots. The higher, the better. (II) Cumulative frequency of the percentage of robots that remained outside the target region at the end of each experimental run. (III) Cumulative frequency of the percentage of robots in the target region that crossed the perimeter.

and turns ineffective; see Fig. 5c (I,II). In fewer cases, `EvoSpace` designed a similar collective behavior to that observed in ANY-POINT CLOSENESS. The robots explore the arena until they step into the target region. Once a robot enters the region, it switches to an in-place rotation behavior. Robots that reach the target region react to the presence of incoming robots that push them further. This behavior is more efficient at placing robots in the target region; see Fig. 5c (III,IV). However, we do not find that the robots are capable of establishing the desired networks.

The ability of the robots to remain at a fixed distance with respect to their peers plays an important role in this mission. To cover a large area, the robots should maximize the relative distance between each other. However, they must constrain this distance to avoid losing the connectivity of the network. Robot swarms designed by `Mate` self-organize in the form of repetitive and cohesive lattices of robots—being effective in the execution of the mission. On the contrary, although the swarms designed by `Chocolate` are capable of establishing small networks, they neither maintain a stable connectivity nor properly cover the target region.

5.3. CONDITIONAL COVERAGE

Fig. 6 (I) shows the numerical results of CONDITIONAL COVERAGE. The control software produced by `Mate` performs significantly better than the one produced by `Chocolate` and `EvoSpace`—both in simulation and reality.

Fig. 6 (II) shows the cumulative frequency of the percentage of robots that remained outside the target region at the end of each experimental run.

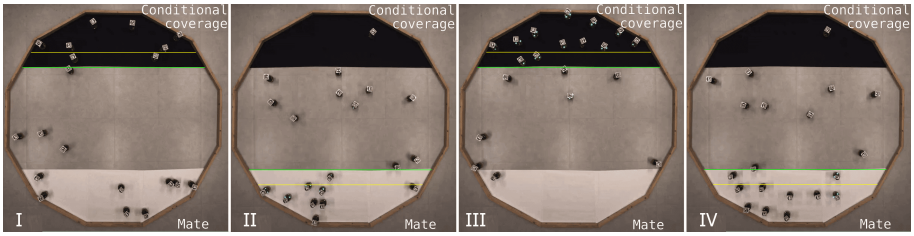
Fig. 6 (III) shows the cumulative frequency of the percentage of robots in the target region that crossed the perimeter. The cumulative frequency plots show that `Chocolate` and `Mate` are less prone than `EvoStick` to leave robots outside the target region at the end of the experiment. `Chocolate` and `Mate` appear to design collective behaviors in which a similar percentage of the robots reach the target region. Likewise NETWORKED COVERAGE, in this mission, the percentage of robots in the target region that cross the perimeter tends to be higher for `Mate` than for `EvoSpace` and `Chocolate`. However, in this mission, `EvoSpace` and `Chocolate` tend to have a similar percentage of robots crossing the perimeter. Fig. 7 shows illustrative examples of the spatial distribution of the robots at the end of four experimental runs of CONDITIONAL COVERAGE.

The robot swarms designed by `Mate` selected and uniformly covered the target region indicated by the initial conditions of the experiment in most cases; see Fig. 7a (III,IV). Yet, in some experimental runs, the robots failed on identifying the appropriate target region or attempted to cover both the black and the white area; see Fig. 7a (I,II). In the more successful runs, as in the two other missions, the robots uniformly distribute in the region in which they arrive (i.e., the target region) by steadily executing FORMATION.

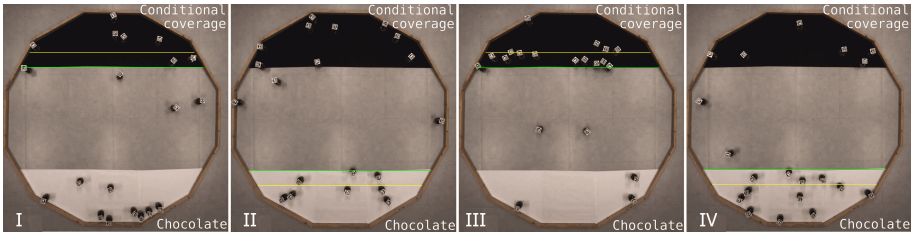
In the case of `Chocolate`, the swarm often fails to select the appropriate target region; see Fig. 7b (I-II). Unfortunately, in the successful runs, the swarm does not display an uniform coverage; see Fig. 7b (III, IV). This behavior is consistent with those observed in the two other missions.

Robots executing the control software produced by `EvoSpace` randomly walk in the arena and do not react when they enter the white or black area. Indeed, the robots seem to disregard the information provided by the color of the floor and/or the interaction with their peers. Consequently, the swarm fails to cover the target region and performs poorly; see Fig. 7c (I–IV). This behavior is consistent across all runs.

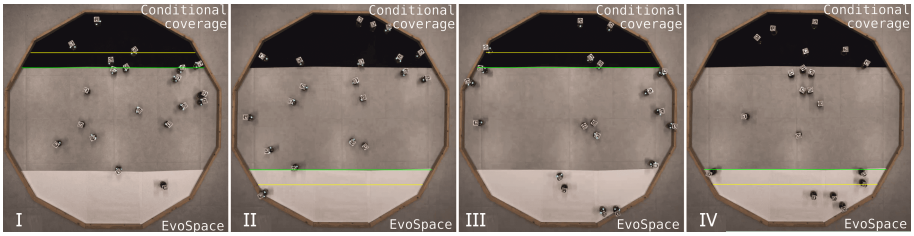
In this mission, FORMATION also contributed to designing spatially-organizing behaviors in which the robots efficiently cover the target region—which ultimately is reflected in the better performance of `Mate`. On the other hand, the performance of `Chocolate` and `EvoSpace` is strongly affected by the individualistic behavior of the robots. As the robots act without considering the presence of their peers, they are less likely to achieve a meaningful spatial distribution. When compared with the other two missions, CONDITIONAL COVERAGE turned to be more challenging for the design methods. As we originally expected, this is probably due to the decision making component



(a) `Mate`



(b) `Chocolate`



(c) `EvoSpace`

Figure 7: Illustrative examples of the final spatial distribution of robots in ANY-POINT CLOSENESS. Each row shows results obtained with four different instances of control software produced by (a) `Mate`, (b) `Chocolate`, and (c) `EvoSpace`. In all cases, the target region in (I,III) is black and the target region in (II,IV) is white. The target region is lined in green and the perimeter is lined in yellow.

of the mission. The percentage of robots that remain outside the target region at the end of the experimental run increased for the three methods. The highest increment was observed in the results obtained by `Chocolate`, and the lowest increment was observed in the results obtained by `Mate`.

5.4. Aggregate results

We first discuss general aspects of the controls software produced by `Mate` and `Chocolate`, and afterwards, we present results on the aggregated performance of `Mate`, `Chocolate` and `EvoSpace` across the three missions.

5.4.1. Control software analysis

Fig. 8 shows heat-map plots that represent the overall usage of the behavior modules available to `Mate` and `Chocolate`. In all cases, we observe that the swarm performs the missions in two phases: an initial phase in which the robots mostly explore the arena—about the first 30 s; and a later phase in which the robots distribute in the space according to the specification of each mission—the remaining time after the first 30 s.

The heat-map plots show that `Mate` produces control software that uses EXPLORATION, PHOTOTAXIS, ANTI-PHOTOTAXIS for exploration, and FORMATION to position the robots. This set of behaviors consistently appears across the three missions. As defined in `Chocolate`, the implementations of ANTI-PHOTOTAXIS and PHOTOTAXIS embed default exploration schemes that execute a ballistic motion in the absence of an ambient light source [14]. The missions we propose do not consider an ambient light source, and therefore, ANTI-PHOTOTAXIS and PHOTOTAXIS operate in practice as random walk behaviors. The robots explore the arena until they identify the target region with the help of the environmental cues—*i.e.*, the color of the floor. We originally expected that the robots would also use ATTRACTION and REPULSION to interact with their peers at large distances, and STOP to adopt steady positions. However, results show that the robots mainly rely on FORMATION to establish the spatially-organizing behaviors specified in each mission.

`Chocolate` produces control software that mostly uses PHOTOTAXIS and ANTI-PHOTOTAXIS for exploration, and STOP to position the robots. This set of behaviors repeatedly appears in the three missions. We expected that `Chocolate`, in the absence of FORMATION, could design collective behaviors in which the robots would continuously transition between ATTRACTION and REPULSION. By doing so, the robots would maintain a rather fixed distance between each other. However, `Chocolate` often converges to a simpler but

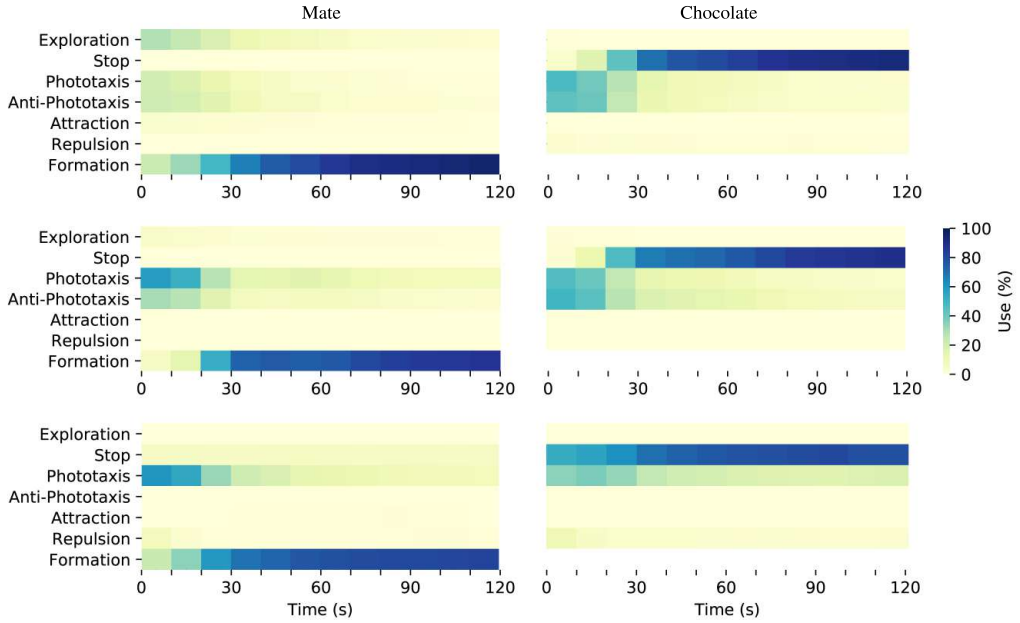


Figure 8: Heat-map plots of the average usage of `Mate`'s (left) and `Chocolate`'s (right) behavior modules in the three missions: ANY-POINT CLOSENESS (top), NETWORKED COVERAGE (middle), and CONDITIONAL COVERAGE (bottom).

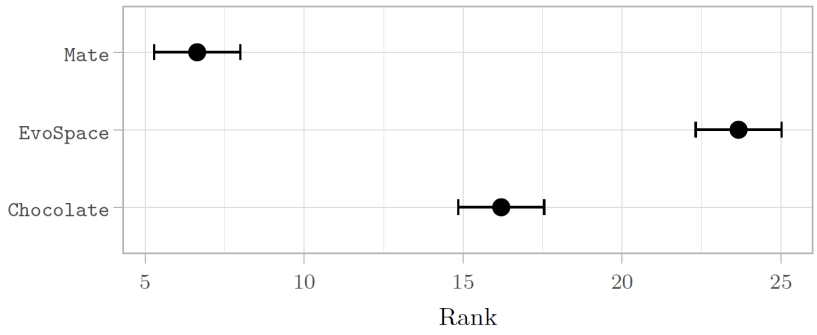


Figure 9: Friedman test on the aggregate results of the three missions. The lower rank, the better.

less effective solution: the robots rely on PHOTOTAXIS, ANTI-PHOTOTAXIS, and the color of the floor to find the target region, and once there, they transition to STOP to maintain a standstill behavior and remain inside.

We argue that by including FORMATION, `Mate` is a more suitable option than `Chocolate` to address missions that require the design of spatially-organizing behaviors with constrains on the distance between robots.

5.4.2. Friedman test and reality gap

Fig. 9 shows the aggregate results of the experiments with physical robots across the three missions. The plot represents the average rank of the three methods and their 95% confidence interval. In our experiments, `Mate` performed significantly better than `Chocolate` and `EvoSpace`. As previously described, `Mate` designed collective behaviors that achieved a better spatial distribution of robots than the baseline methods. The class of missions we studied highly depends on such ability, and therefore, `Mate` outranked the two baseline methods.

We observed the effects of the reality gap while assessing the control software produced by the three methods: in the three missions, there was a performance drop when the control software was ported from the simulation to the e-pucks. The reality gap manifested itself differently for the design methods—`Mate` and `Chocolate` were less prone to suffer its effects in comparison to `EvoSpace`. These results are consistent with those reported in other studies that compare the modular and the neuro-evolutionary approaches to the automatic off-line design of robot swarms [12, 22, 23].

As originally noticed by Francesca *et al.* [11], automatic design methods appear to face a problem that is similar to the generalization problem [82] encountered in machine learning. That is, design methods with a high representational power can yield a sort of *overfitting* to the simulation environment—which then typically hinders the performance of the control software when assessed with physical robots [22, 34]. In this context, it is expected that methods that differ in their representational power might also overfit to different extent the simulation and, therefore, show a different drop in performance due to the effects of the reality gap [22]. Neuro-evolutionary methods have a higher representational power than modular methods. In neuro-evolutionary methods, the artificial neural networks and evolutionary algorithms are known to have a high representational power that exploits the idiosyncrasies of the simulator to produce high-performing solutions [83, 84]. A large body of the literature has previously reported this phenomenon and has shown that control software evolved in simulation most often performs well in simulation but badly when ported to physical robots [23, 85]. As shown in our experiments, `EvoSpace` was capable of producing control software that performs well in simulation but that suffered a large performance drop in reality. The performance drop of `EvoSpace` is considerably larger than the one observed in `Mate` and `Chocolate`—although the methods are tested under the same experimental conditions. The modular nature of `Mate` and `Chocolate` intrinsically limits the representational power of the design methods: these methods can only produce the control software that results from the combination of the given parametric software modules. The space of possible control software is considerably smaller in `Mate` and `Chocolate` than the one that results from the implementation of the artificial neural network in `EvoSpace`. In consequence, we argue that `Mate` and `Chocolate` are less prone to overfit the simulation environment and, therefore, they suffer less from the effects of the reality gap when ported to the physical robots. For a more detailed discussion on the effects of the reality gap in modular and neuro-evolutionary methods, see [22, 23, 34].

In a few experimental runs executed with `Mate`, one or two robots malfunctioned because they either (i) got stuck due to imperfections in the floor of the arena or because (ii) they run out of battery. Still, in all cases, the swarm remained operative, achieved the mission at hand, and did not considerably decrease its performance. These observations suggest that the control software produced by `Mate` might not be severely affected in case of failure of individual robots. Previous studies conducted with AutoMoDe methods have

reported being robust against changes in the number of operative robots [14]. The observations made in our experiments suggest that this property might also be a property of Mate. However, further experimentation is required to corroborate this conjecture. Examples of the aforementioned issues can be observed in the videos we provide as supplementary material [81].

6. Conclusions

We studied the design of spatially-organizing behaviors in the context of the automatic off-line design of robot swarms. To this purpose, we presented Mate: a modular method conceived to address missions in which the robot swarm must operate under spatial distribution constrains. With Mate, we introduced FORMATION: a behavior module that enables the stable formation of robots in hexagonal patterns.

We assessed Mate in simulation and with physical e-puck robots in three missions: ANY-POINT CLOSENESS, NETWORKED COVERAGE and CONDITIONAL COVERAGE. In these missions, the robots must operate while considering environmental cues and the relative distance that they maintain with respect to their peers. The control software designed by Mate resulted effective in the three missions. In ANY-POINT CLOSENESS, the robots uniformly covered an indicated target region in the arena. In NETWORKED COVERAGE, the robots established coverage networks while maintaining connectivity at a fixed distance. Finally, in CONDITIONAL COVERAGE, Mate designed collective behaviors that allowed the swarm to selectively cover one out of two possible target regions.

We compared Mate with two baseline methods: Chocolate—a state-of-the-art modular method; and EvoSpace—a method of the neuro-evolutionary approach. The aggregated results show that the control software produced by Mate performs significantly better than the one produced by the baseline methods. Our study also evaluated the portability of the control software produced by Mate from the simulation to the physical robots. In this case, our results are in-line with recent findings that illustrate how modular methods are more robust to the effects of the reality gap than the neuro-evolutionary approach. Indeed, the performance drop was smaller in Mate and Chocolate than in EvoSpace.

Initially, we expected that Chocolate and EvoSpace would be capable of addressing the missions that we conceived to assess Mate. Chocolate includes behaviors modules that could emulate the effects of FORMATION. Yet,

the collective behaviors designed by Chocolate did not exhibit the spatial distribution properties observed in those designed by Mate. In a like manner, EvoSpace incorporates the model of virtual physics that we used to conceive FORMATION. Yet, we did not notice any meaningful use of this information in the behavior displayed by the robots. We conclude that the inability of Chocolate and EvoSpace to design behaviors in which robots maintain specific relative positions with respect to each other translated in a lower performance.

We argue that the aforementioned results highlight limitations of existent automatic methods for the design of spatially-organizing behaviors for robot swarms. First, although design methods based on neuro-evolution have been used to design spatially-organizing behaviors in the past (*e.g.,* pattern-formation and flocking), they strongly suffer from the effects of the reality gap. Second, promising alternative approaches like the modular design do not provide yet the means to design complex spatially-organizing behaviors with particular robot positioning schemes. For example, existing modular design methods (including Mate) cannot produce control software for the organization of robot swarms in complex patterns or chains of robots.

By introducing Mate, we go a step further in that direction. In this paper, we focused on demonstrating that automatic modular design can effectively address missions that require a specific and constrained spatial organization of the robots in the environment. We will devote future work to enlarge the number of parameters that are fine-tuned in Mate, which could enable the design of new and more complex spatially-organizing behaviors. For example, we plan to address classes of missions related to the formation of chains of robots and the formation of swarms that move in patterns. Also, we wish to investigate the effects of robot malfunctioning on the stability of the spatially-organizing behaviors designed by Mate, and propose strategies to mitigate its effects.

As of today, most research in swarm robotics has been conducted with manual design methods that target a single platform and that focus on a specific mission. Recent perspectives on the future of swarm robotics endorse the conception of design methodologies that facilitate sharing and replicating experiments with different robot platforms [6]. Certainly, an important open issue in automatic design is how to conceive design methods like Mate and make them portable to a wide class of robots. The reference model introduced with AutoMoDe is, in a sense, a level of abstraction with respect to the underlying robotic platform that can facilitate the portability of the methods.

In this paper, we limited the scope of the study to the design of spatially-organizing behaviors for a version of the e-puck that is compliant with the reference model RM 3.1. In principle, `Mate` can be adapted to produce control software for other robots that share the same functional capabilities defined in this reference model. We believe that conceiving design methods that target robots that are formally defined by a reference model is a first step to enable portability and ease the replication of experiments. In our future work, we will investigate how to extend further this idea within the context of the automatic design of robot swarms.

Ethical approval

Not applicable.

Consent to Participate

Not applicable.

Consent to Publish

Not applicable.

Funding

The project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (DEMIURGE Project, grant agreement No 681872) and from Belgium's Wallonia-Brussels Federation through the ARC Advanced project GbO-Guaranteed by Optimization. MB acknowledges support from the Belgian Fonds de la Recherche Scientifique – FNRS. DGR acknowledges support from the Colombian Ministry of Science, Technology and Innovation – Minciencias. FJM, AMNL and MRAM acknowledge support from the Coordination for the Improvement of Higher Education Personnel (CAPES-Brazil) and the National Council for Scientific and Technological Development (CNPq-Brazil).

Availability of data and materials

The control software produced during the experiments, associated demonstrative videos with physical robots, and pictures of the final positioning of the robots for each experimental run are available online at `https://iridia.ulb.ac.be/supp/IridiaSupp2020-008`.

The source code is available as free and open-source. Otherwise indicated, the software is available under the MIT License in the following repositories: (i) ARGoS3-AutoMoDe for the implementation of `Mate`[3] and `Chocolate`[4]; (ii) ARGoS3-NEAT for the implementation of `EvoSpace`[5]; (iii) ARGoS3[6] for the ARGoS3 simulator; (iv) argos3-epuck[7] for the ARGoS3 plugin to simulate and operate the e-puck; (v) demiurge-epuck-dao[8] for the software interface that enables the operation of the e-puck with `Mate`, `Chocolate` and `EvoSpace`; (vi) experiments-loop-functions[9] for the implementations of ANY-POINT CLOSENESS, NETWORKED COVERAGE, and CONDITIONAL COVERAGE; (vii) and irace[10] for the implementation of Iterated F-race (GNU General Public License).

CRediT authorship contribution statement

Fernando J. Mendiburu: Conceptualization, Methodology, Software, Validation, Investigation, Data curation, Writing - original draft, Writing - review & editing, Visualization. **David Garzón Ramos**: Conceptualization, Methodology, Validation, Investigation, Writing - original draft, Writing - review & editing, Visualization. **Marcos R.A. Morais**: Methodology. **Antonio M.N. Lima**: Methodology. **Mauro Birattari**: Conceptualization, Methodology, Writing - review & editing, Supervision.

[3] `https://doi.org/10.5281/zenodo.5893277`
[4] `https://doi.org/10.5281/zenodo.4849541`
[5] `https://doi.org/10.5281/zenodo.4849517`
[6] `https://doi.org/10.5281/zenodo.4889111`
[7] `https://doi.org/10.5281/zenodo.4882714`
[8] `https://doi.org/10.5281/zenodo.5893406`
[9] `https://doi.org/10.5281/zenodo.5893411`
[10] `https://doi.org/10.5281/zenodo.4888996`

Declaration of Competing Interest

The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

[1] G. Beni, From swarm intelligence to swarm robotics, in: E. Şahin, W. M. Spears (Eds.), Swarm Robotics, SAB, Vol. 3342 of LNCS, Springer, Berlin, Germany, 2004, pp. 1–9. doi:10.1007/978-3-540-30552-1_1.

[2] E. Şahin, Swarm robotics: from sources of inspiration to domains of application, in: E. Şahin, W. M. Spears (Eds.), Swarm Robotics, SAB, Vol. 3342 of LNCS, Springer, Berlin, Germany, 2004, pp. 10–20. doi:10.1007/978-3-540-30552-1_2.

[3] M. Brambilla, E. Ferrante, M. Birattari, M. Dorigo, Swarm robotics: a review from the swarm engineering perspective, Swarm Intelligence 7 (1) (2013) 1–41. doi:10.1007/s11721-012-0075-2.

[4] M. Dorigo, M. Birattari, M. Brambilla, Swarm robotics, Scholarpedia 9 (1) (2014) 1463. doi:10.4249/scholarpedia.1463.

[5] M. Dorigo, G. Theraulaz, V. Trianni, Reflections on the future of swarm robotics, Science Robotics 5 (2020) eabe4385. doi:10.1126/scirobotics.abe4385.

[6] M. Dorigo, G. Theraulaz, V. Trianni, Swarm robotics: past, present, and future [point of view], Proceedings of the IEEE 109 (7) (2021) 1152–1165. doi:10.1109/JPROC.2021.3072740.

[7] H. Hamann, M. Schranz, W. Elmenreich, V. Trianni, C. Pinciroli, N. Bredeche, E. Ferrante, Editorial: designing self-organization in the physical realm, Frontiers in Robotics and AI 7 (2020) 164. doi:10.3389/frobt.2020.597859.

[8] M. Birattari, A. Ligot, D. Bozhinoski, M. Brambilla, G. Francesca, L. Garattoni, D. Garzón Ramos, K. Hasselmann, M. Kegeleirs, J. Kuckling, F. Pagnozzi, A. Roli, M. Salman, T. Stützle, Automatic off-line

design of robot swarms: a manifesto, Frontiers in Robotics and AI 6 (2019) 59. doi:10.3389/frobt.2019.00059.

[9] S. Nolfi, D. Floreano, Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines, 1st Edition, MIT Press, Cambridge, MA, USA, 2000, a Bradford Book.

[10] V. Trianni, Evolutionary Swarm Robotics, Springer, Berlin, Germany, 2008. doi:10.1007/978-3-540-77612-3.

[11] G. Francesca, M. Brambilla, A. Brutschy, V. Trianni, M. Birattari, AutoMoDe: a novel approach to the automatic design of control software for robot swarms, Swarm Intelligence 8 (2) (2014) 89–112. doi:10.1007/s11721-014-0092-4.

[12] G. Francesca, M. Brambilla, A. Brutschy, L. Garattoni, R. Miletitch, G. Podevijn, A. Reina, T. Soleymani, M. Salvaro, C. Pinciroli, F. Mascia, V. Trianni, M. Birattari, AutoMoDe-Chocolate: automatic design of control software for robot swarms, Swarm Intelligence 9 (2–3) (2015) 125–152. doi:10.1007/s11721-015-0107-9.

[13] M. Salman, A. Ligot, M. Birattari, Concurrent design of control software and configuration of hardware for robot swarms under economic constraints, PeerJ Computer Science 5 (2019) e221. doi:10.7717/peerj-cs.221.

[14] G. Spaey, M. Kegeleirs, D. Garzón Ramos, M. Birattari, Evaluation of alternative exploration schemes in the automatic modular design of robot swarms, in: B. Bogaerts, G. Bontempi, P. Geurts, N. Harley, B. Lebichot, T. Lenaerts, G. Louppe (Eds.), Artificial Intelligence and Machine Learning: BNAIC 2019, BENELEARN 2019, Vol. 1196 of CCIS, Springer, Cham, Switzerland, 2020, pp. 18–33. doi:10.1007/978-3-030-65154-1_2.

[15] K. Hasselmann, M. Birattari, Modular automatic design of collective behaviors for robots endowed with local communication capabilities, PeerJ Computer Science 6 (2020) e291. doi:10.7717/peerj-cs.291.

[16] J. Kuckling, K. Ubeda Arriaza, M. Birattari, AutoMoDe-IcePop: automatic modular design of control software for robot swarms using simulated annealing, in: B. Bogaerts, G. Bontempi, P. Geurts, N. Harley,

B. Lebichot, T. Lenaerts, G. Louppe (Eds.), Artificial Intelligence and Machine Learning: BNAIC 2019, BENELEARN 2019, Vol. 1196 of CCIS, Springer, Cham, Switzerland, 2020, pp. 3–17.

[17] D. Garzón Ramos, M. Birattari, Automatic design of collective behaviors for robots that can display and perceive colors, Applied Sciences 10 (13) (2020) 4654. doi:10.3390/app10134654.

[18] A. Ligot, K. Hasselmann, M. Birattari, AutoMoDe-Arlequin: neural networks as behavioral modules for the automatic design of probabilistic finite state machines, in: M. Dorigo, T. Stützle, M. J. Blesa, C. Blum, H. Hamann, M. K. Heinrich, V. Strobel (Eds.), Swarm Intelligence – ANTS, Vol. 12421 of LNCS, Springer, Cham, Switzerland, 2020, pp. 109–122. doi:10.1007/978-3-030-60376-2_21.

[19] F. Pagnozzi, M. Birattari, Off-policy evaluation of the performance of a robot swarm: Importance sampling to assess potential modifications to the finite-state machine that controls the robots, Frontiers in Robotics and AI 8 (2021) 55. doi:10.3389/frobt.2021.625125.

[20] A. Ligot, J. Kuckling, D. Bozhinoski, M. Birattari, Automatic modular design of robot swarms using behavior trees as a control architecture, PeerJ Computer Science 6 (2020) e314. doi:10.7717/peerj-cs.314.

[21] J. Kuckling, T. Stützle, M. Birattari, Iterative improvement in the automatic modular design of robot swarms, PeerJ Computer Science 6 (2020) e322. doi:10.7717/peerj-cs.322.

[22] A. Ligot, M. Birattari, Simulation-only experiments to mimic the effects of the reality gap in the automatic design of robot swarms, Swarm Intelligence (2019) 1–24doi:10.1007/s11721-019-00175-w.

[23] K. Hasselmann, A. Ligot, J. Ruddick, M. Birattari, Empirical assessment and comparison of neuro-evolutionary methods for the automatic offline design of robot swarms, Nature Communications 12 (2021) 4345. doi:10.1038/s41467-021-24642-3.

[24] L. Garattoni, M. Birattari, Swarm robotics, in: J. G. Webster (Ed.), Wiley Encyclopedia of Electrical and Electronics Engineering, John Wiley & Sons, Hoboken, NJ, USA, 2016, pp. 1–19. doi:10.1002/047134608X.W8312.

[25] M. Gauci, J. Chen, W. Li, T. J. Dodd, R. Groß, Self-organized aggregation without computation, The International Journal of Robotics Research 33 (8) (2014) 1145–1161. doi:10.1177/0278364914525244.

[26] L. Garattoni, M. Birattari, Autonomous task sequencing in a robot swarm, Science Robotics 3 (20) (2018) eaat0430. doi:10.1126/scirobotics.aat0430.

[27] S. Nouyan, A. Campo, M. Dorigo, Path formation in a robot swarm: self-organized strategies to find your way home, Swarm Intelligence 2 (1) (2008) 1–23. doi:10.1007/s11721-007-0009-6.

[28] M. Rubenstein, A. Cornejo, R. Nagpal, Programmable self-assembly in a thousand-robot swarm, Science 345 (6198) (2014) 795–799. doi:10.1126/science.1254295.

[29] M. Dorigo, D. Floreano, L. M. Gambardella, F. Mondada, S. Nolfi, T. Baaboura, M. Birattari, M. Bonani, M. Brambilla, A. Brutschy, D. Burnier, A. Campo, A. L. Christensen, A. Decugnière, G. A. Di Caro, F. Ducatelle, E. Ferrante, A. Förster, J. Martinez Gonzales, J. Guzzi, V. Longchamp, S. Magnenat, N. Mathews, M. Montes de Oca, R. O'Grady, C. Pinciroli, G. Pini, P. Retornaz, J. Roberts, V. Sperati, T. Stirling, A. Stranieri, T. Stützle, V. Trianni, E. Tuci, A. E. Turgut, F. Vaussard, Swarmanoid: a novel concept for the study of heterogeneous robotic swarms, IEEE Robotics & Automation Magazine 20 (4) (2013) 60–71. doi:10.1109/MRA.2013.2252996.

[30] M. Gauci, J. Chen, W. Li, T. J. Dodd, R. Groß, Clustering objects with robots that do not compute, in: Proceedings of the 2014 International Conference on Autonomous Agents and Multi-Agent Systems – AAMAS2014, AAMAS '14, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, USA, 2014, pp. 421–428. doi:10.5555/2615731.2615800.

[31] J. Werfel, K. Petersen, R. Nagpal, Designing collective behavior in a termite-inspired robot construction team, Science 343 (6172) (2014) 754–758. doi:10.1126/science.1245842.

[32] W. M. Spears, D. Spears, R. Heil, W. Kerr, S. Hettiarachchi, An overview of physicomimetics, in: E. Şahin, W. M. Spears (Eds.), Swarm Robotics, SAB, Springer, Berlin, Heidelberg, Germany, 2005, pp. 84–97.

[33] M. Duarte, V. Costa, J. Gomes, T. Rodrigues, F. Silva, S. M. Oliveira, A. L. Christensen, Evolution of collective behaviors for a real swarm of aquatic surface robots, PLOS ONE 11 (3) (2016) e0151834. doi:10.1371/journal.pone.0151834.

[34] M. Birattari, A. Ligot, G. Francesca, AutoMoDe: a modular approach to the automatic off-line design and fine-tuning of control software for robot swarms, in: N. Pillay, R. Qu (Eds.), Automated Design of Machine Learning and Search Algorithms, Natural Computing Series, Springer, Cham, Switzerland, 2021, pp. 73–90. doi:10.1007/978-3-030-72069-8_5.

[35] W. M. Spears, D. F. Gordon, Using artificial physics to control agents, in: Proceedings 1999 International Conference on Information Intelligence and Systems, IEEE Computer Society Press, Los Alamitos, CA, USA, 1999, pp. 281–288. doi:10.1109/ICIIS.1999.810278.

[36] J. E. Jones, On the determination of molecular fields, Proceedings of the Royal Society of London 106 (1924) 463–477.

[37] R. Olfati-Saber, Flocking for multi-agent dynamic systems: algorithms and theory, IEEE Transactions on Automatic Control 51 (3) (2006) 401–420. doi:10.1109/TAC.2005.864190.

[38] M. J. Matarić, Interaction and intelligent behavior, Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, MA, USA (1994).

[39] T. Balch, R. C. Arkin, Behavior-based formation control for multirobot teams, IEEE Transactions on Robotics and Automation 14 (6) (1998) 926–939.

[40] M. Birattari, A. Ligot, K. Hasselmann, Disentangling automatic and semi-automatic approaches to the optimization-based design of control software for robot swarms, Nature Machine Intelligence 2 (9) (2020) 494–499. doi:10.1038/s42256-020-0215-0.

[41] Y. K. Lopes, S. M. Trenkwalder, A. B. Leal, T. J. Dodd, R. Groß, Supervisory control theory applied to swarm robotics, Swarm Intelligence 10 (1) (2016) 65–97. doi:10.1007/s11721-016-0119-0.

[42] Y. K. Lopes, S. M. Trenkwalder, A. B. Leal, T. J. Dodd, R. Groß, Probabilistic supervisory control theory (pSCT) applied to swarm robotics,

in: Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems – AAMAS2017, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, USA, 2017, pp. 1395–1403.

[43] Y. K. Lopes, A. B. Leal, T. J. Dodd, R. Groß, Application of supervisory control theory to swarms of e-puck and Kilobot robots, in: M. Dorigo, M. Birattari, S. Garnier, H. Hamann, M. Montes de Oca, C. Solnon, T. Stützle (Eds.), Swarm Intelligence – ANTS, Vol. 8667 of LNCS, Springer, Cham, Switzerland, 2014, pp. 62–73. doi:10.1007/978-3-319-09952-1_6.

[44] O. Khatib, Real-time obstacle avoidance for manipulators and mobile robots, The International Journal of Robotics Research 5 (1) (1986) 90–98.

[45] W. M. Spears, D. Spears, J. C. Hamann, R. Heil, Distributed, physics-based control of swarms of vehicles, Autonomous Robots 17 (2) (2004) 137–162. doi:10.1023/B:AURO.0000033970.96785.f2.

[46] B. Shucker, J. K. Bennett, Scalable control of distributed robotic macrosensors, in: R. Alami, R. Chatila, H. Asama (Eds.), Distributed Autonomous Robotic Systems, Springer, Tokyo, Japan, 2007, pp. 379–388. doi:10.1007/978-4-431-35873-2_37.

[47] T. Lochmatter, E. Aydın Göl, I. Navarro, A. Martinoli, A plume tracking algorithm based on crosswind formations, in: A. Martinoli, F. Mondada, N. Correll, G. Mermoud, M. Egerstedt, M. A. Hsieh, L. E. Parker, K. Støy (Eds.), Distributed Autonomous Robotic Systems: The 10th International Symposium, Springer, Berlin, Germany, 2013, pp. 91–102. doi:10.1007/978-3-642-32723-0_7.

[48] A. Howard, M. J. Matarić, G. S. Sukhatme, Mobile sensor network deployment using potential fields: a distributed, scalable solution to the area coverage problem, in: H. Asama, T. Fukuda, T. Hasegawa (Eds.), Distributed Autonomous Robotic Systems, Springer, Tokyo, Japan, 2002, pp. 299–308. doi:10.1007/978-4-431-65941-9_30.

[49] A. E. Turgut, H. Çelikkanat, F. Gökçe, E. Şahin, Self-organized flocking in mobile robot swarms, Swarm Intelligence 2 (2) (2008) 97–120.

[50] F. J. Mendiburu, M. R. A. Morais, A. M. Nogueira Lima, Behavior coordination in multi-robot systems, in: 2016 IEEE International Conference on Automatica (ICA-ACCA), IEEE, Curico, Chile, 2016, pp. 1–7. doi:10.1109/ICA-ACCA.2016.7778506.

[51] E. Ferrante, A. E. Turgut, C. Huepe, A. Stranieri, C. Pinciroli, M. Dorigo, Self-organized flocking with a mobile robot swarm: a novel motion control method, Adaptive Behavior 20 (6) (2012) 460–477.

[52] G. Francesca, M. Birattari, Automatic design of robot swarms: achievements and challenges, Frontiers in Robotics and AI 3 (29) (2016) 1–9. doi:10.3389/frobt.2016.00029.

[53] M. Duarte, S. M. Oliveira, A. L. Christensen, Evolution of hybrid robotic controllers for complex tasks, Journal of Intelligent & Robotic Systems 78 (3) (2015) 463–484.

[54] M. Duarte, S. M. Oliveira, A. L. Christensen, Hybrid control for large swarms of aquatic drones, in: H. Sayama, J. Rieffel, S. Risi, R. Doursat, H. Lipson (Eds.), Artificial Life 14. Proceedings of the Fourteenth International Conference on the Synthesis and Simulation of Living Systems, MIT Press, Cambridge, MA, USA, 2014, pp. 785–792. doi:10.7551/978-0-262-32621-6-ch105.

[55] M. Duarte, J. Gomes, V. Costa, S. M. Oliveira, A. L. Christensen, Hybrid control for a real swarm robotics system in an intruder detection task, in: G. Squillero, P. Burelli (Eds.), Applications of Evolutionary Computation, 19th European Conference, EvoApplications 2016, Vol. 9598 of Lecture Notes in Computer Science, Springer International Publishing, Cham, Switzerland, 2016, pp. 213–230. doi:10.1007/978-3-319-31153-1_15.

[56] S. Hettiarachchi, W. M. Spears, Distributed adaptive swarm for obstacle avoidance, International Journal of Intelligent Computing and Cybernetics 2 (4) (2009) 644–671.

[57] S. Hettiarachchi, W. M. Spears, DAEDALUS for agents with obstructed perception, in: IEEE Mountain Workshop on Adaptive and Learning Systems, IEEE, Logan, UT, USA, 2006, pp. 195–200.

[58] S. Hettiarachchi, W. M. Spears, Moving swarm formations through obstacle fields, in: Proceedings of the 2005 International Conference on Artificial Intelligence, ICAI'05, Vol. 1, Las Vegas, NV, USA, 2005, pp. 97–103.

[59] C. Pinciroli, M. Birattari, E. Tuci, M. Dorigo, M. del Rey Zapatero, T. Vinko, D. Izzo, Self-organizing and scalable shape formation for a swarm of pico satellites, in: 2008 NASA/ESA Conference on Adaptive Hardware and Systems, IEEE, Noordwijk, Netherlands, 2008, pp. 57–61.

[60] C. Pinciroli, M. Birattari, E. Tuci, M. Dorigo, M. del Rey Zapatero, T. Vinko, D. Izzo, Lattice formation in space for a swarm of pico satellites, in: M. Dorigo, M. Birattari, C. Blum, M. Clerc, T. Stützle, A. Winfield (Eds.), Ant Colony Optimization and Swarm Intelligence – ANTS 2008, Springer, Berlin, Germany, 2008, pp. 347–354. doi:10.1007/978-3-540-87527-7_36.

[61] R. A. Brooks, Artificial life and real robots, in: Proceedings of the First European Conference on Artificial Life, MIT Press, 1992, pp. 3–10.

[62] D. Floreano, P. Husbands, S. Nolfi, Evolutionary robotics, in: B. Siciliano, O. Khatib (Eds.), Springer Handbook of Robotics, Springer Handbooks, Springer, Berlin, Heidelberg, Germany, 2008, pp. 1423–1451, first edition. doi:10.1007/978-3-540-30301-5_62.

[63] F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz, S. Magnenat, J.-C. Zufferey, D. Floreano, A. Martinoli, The e-puck, a robot designed for education in engineering, in: P. Gonçalves, P. Torres, C. Alves (Eds.), Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions, Instituto Politécnico de Castelo Branco, Castelo Branco, Portugal, 2009, pp. 59–65.

[64] W. Liu, A. Winfield, Open-hardware e-puck Linux extension board for experimental swarm robotics research, Microprocessors and Microsystems - Embedded Hardware Design 35 (2011) 60–67. doi:10.1016/j.micpro.2010.08.002.

[65] Á. Gutiérrez, A. Campo, M. Dorigo, J. Donate, F. Monasterio-Huelin, L. Magdalena, Open e-puck range & bearing miniaturized board for local communication in swarm robotics, in: K. Ko-

suge (Ed.), IEEE International Conference on Robotics and Automation, ICRA, IEEE, Piscataway, NJ, USA, 2009, pp. 3111–3116. doi:10.1109/ROBOT.2009.5152456.

[66] École polytechnique fédérale de Lausanne, Omnidirectional vision turret for the e-puck, http://www.e-puck.org/index.php?option=com_content&view=article&id=26&Itemid=21 (2010).

[67] K. Hasselmann, A. Ligot, G. Francesca, D. Garzón Ramos, M. Salman, J. Kuckling, F. J. Mendiburu, M. Birattari, Reference models for AutoMoDe, Tech. Rep. TR/IRIDIA/2018-002, IRIDIA, Université libre de Bruxelles, Belgium (2018).

[68] D. Spears, D. Thayer, D. V. Zarzhitsky, A multi-robot chemical source localization strategy based on fluid physics: theoretical principles, in: W. M. Spears, D. Spears (Eds.), Physicomimetics: Physics-Based Swarm Intelligence, Springer, Berlin, Germany, 2012, pp. 223–249. doi:10.1007/978-3-642-22804-9.

[69] T. Apker, M. A. Potter, Physicomimetic motion control of physically constrained agents, in: W. M. Spears, D. Spears (Eds.), Physicomimetics: Physics-Based Swarm Intelligence, Springer, Berlin, Germany, 2012, pp. 413–437. doi:10.1007/978-3-642-22804-9.

[70] J. Kellogg, C. Bovais, J. Dahlburg, R. J. Foch, J. H. Gardner, D. F. Gordon, R. L. Hartley, B. Kamgar-Parsi, H. Mcfarlane, F. Pipitone, R. Ramamurti, A. Sciambi, W. M. Spears, D. Srull, C. Sullivan, The NRL micro tactical expendable (MITE) air vehicle, The Aeronautical Journal (2002) 431–441.

[71] M. López-Ibáñez, J. Dubois-Lacoste, L. Pérez Cáceres, M. Birattari, T. Stützle, The irace package: iterated racing for automatic algorithm configuration, Operations Research Perspectives 3 (2016) 43–58. doi:10.1016/j.orp.2016.09.002.

[72] W. J. Conover, Practical Nonparametric Statistics, 3rd Edition, Wiley series in probability and statistics, applied probability and statistics section, John Wiley & Sons, New York, NY, USA, 1999.

[73] G. Francesca, M. Brambilla, A. Brutschy, L. Garattoni, R. Miletitch, G. Podevijn, A. Reina, T. Soleymani, M. Salvaro, C. Pinciroli, V. Trianni, M. Birattari, An experiment in automatic design of robot swarms: AutoMoDe-Vanilla, EvoStick, and human experts, in: M. Dorigo, M. Birattari, S. Garnier, H. Hamann, M. Montes de Oca, C. Solnon, T. Stützle (Eds.), Swarm Intelligence – ANTS, Vol. 8667 of LNCS, Springer International Publishing, Cham, Switzerland, 2014, pp. 25–37. doi:10.1007/978-3-319-09952-1_3.

[74] N. Hansen, A. Ostermeier, Completely derandomized self-adaptation in evolution strategies, Evolutionary Computation 9 (2) (2001) 159–195. doi:10.1162/106365601750190398.

[75] T. Glasmachers, T. Schaul, S. Yi, D. Wierstra, J. Schmidhuber, Exponential natural evolution strategies, in: Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation, GECCO, ACM, 2010, pp. 393–400. doi:10.1145/1830483.1830557.

[76] K. O. Stanley, R. Miikkulainen, Evolving neural networks through augmenting topologies, Evolutionary Computation 10 (2) (2002) 99–127. doi:10.1162/106365602320169811.

[77] C. Pinciroli, V. Trianni, R. O'Grady, G. Pini, A. Brutschy, M. Brambilla, N. Mathews, E. Ferrante, G. A. Di Caro, F. Ducatelle, M. Birattari, L. M. Gambardella, M. Dorigo, ARGoS: a modular, parallel, multi-engine simulator for multi-robot systems, Swarm Intelligence 6 (4) (2012) 271–295. doi:10.1007/s11721-012-0072-5.

[78] L. Garattoni, G. Francesca, A. Brutschy, C. Pinciroli, M. Birattari, Software infrastructure for e-puck (and TAM), Tech. Rep. TR/IRIDIA/2015-004, IRIDIA, Université libre de Bruxelles, Belgium (2015).

[79] L. Pitonakova, M. Giuliani, A. Pipe, A. Winfield, Feature and performance comparison of the V-REP, Gazebo and ARGoS robot simulators, in: M. Giuliani, T. Assaf, M. E. Giannaccini (Eds.), Towards Autonomous Robotic Systems, LNCS, Springer, Cham, Switzerland, 2018, pp. 357–368. doi:10.1007/978-3-319-96728-8_30.

[80] A. Stranieri, A. E. Turgut, M. Salvaro, L. Garattoni, G. Francesca, A. Reina, M. Dorigo, M. Birattari, IRIDIA's arena tracking system, Tech. Rep. TR/IRIDIA/2013-013, IRIDIA, Université libre de Bruxelles, Belgium (2013).

[81] F. J. Mendiburu, D. Garzón Ramos, M. R. A. Morais, A. M. Nogueira Lima, M. Birattari, AutoMoDe-`Mate`: automatic off-line design of spatially-organizing behaviors for robot swarms: supplementary material, `https://iridia.ulb.ac.be/supp/IridiaSupp2020-008` (2020).

[82] S. Geman, E. Bienenstock, R. Doursat, Neural networks and the bias/variance dilemma, Neural Computation 4 (1) (1992) 1–58. doi:10.1162/neco.1992.4.1.1.

[83] A. Eiben, Real-world robot evolution: why would it (not) work?, Frontiers in Robotics and AI 8 (2021) 243. doi:10.3389/frobt.2021.696452.

[84] S. Doncieux, N. Bredeche, J.-B. Mouret, A. Eiben, Evolutionary robotics: what, why, and where to, Frontiers in Robotics and AI 2 (2015) 4. doi:10.3389/frobt.2015.00004.

[85] F. Van Diggelen, E. Ferrante, N. Harrak, J. Luo, D. Zeeuwe, A. Eiben, The influence of robot traits and evolutionary dynamics on the reality gap, IEEE Transactions on Cognitive and Developmental Systems (2021) 1doi:10.1109/TCDS.2021.3112236.