# Modular design of control software for robot swarms

The collective behavior of a robot swarm—and hence its ability to accomplish a particular mission—results from the interactions that the robots have with the environment and with their peers [1]. Unfortunately, conceiving and implementing a collective behavior for a robot swarm is particularly challenging. The desired collective behavior for the robots is specified globally for the swarm, but the robots must be programmed individually. The challenge is that no generally applicable method exists to tell what an individual robot should do so that the desired collective behavior is obtained in the swarm [2].

In this practical session, the students will use a modular approach to design control software for a swarm of 20 e-puck robots that must perform a set of missions. The missions require the robots to communicate, navigate the environment, react to events, and display spatial-organization properties.

The exercise is based on `TuttiFrutti` [3], a modular design method specialized in the realization of collective behaviors for robots that can display and perceive color signals. `TuttiFrutti` generates control software in the form of probabilistic finite-state machines that combine parametric software modules. In `TuttiFrutti`, the design process is conducted by an optimization algorithm that searches the space of possible control software for good-performing instances. Conversely, in this practical session, the students will take on the role of optimization agents, combining and tuning `TuttiFrutti`'s software modules to create good-performing control software for the robots. The goal of this practical session is therefore to demonstrate how parametric software modules can be combined in different ways to obtain a variety of collective behaviors with a robot swarm.

To perform the exercise, students are provided with a visualization tool [4] (i) to produce and manipulate finite-state machines, (ii) to visualize simulations [5] of the resulting collective behavior, and (iii) to compute the performance of the swarm in each mission.

## 1 Installing the software

Three options are available to install the required software.

### 1.1 Docker users

The software has been pre-installed in the Docker image of the Summer School. Instructions for pulling the image are available at:

```
https://github.com/IntelligentRoboticsLabs/docker_infrastructure
```

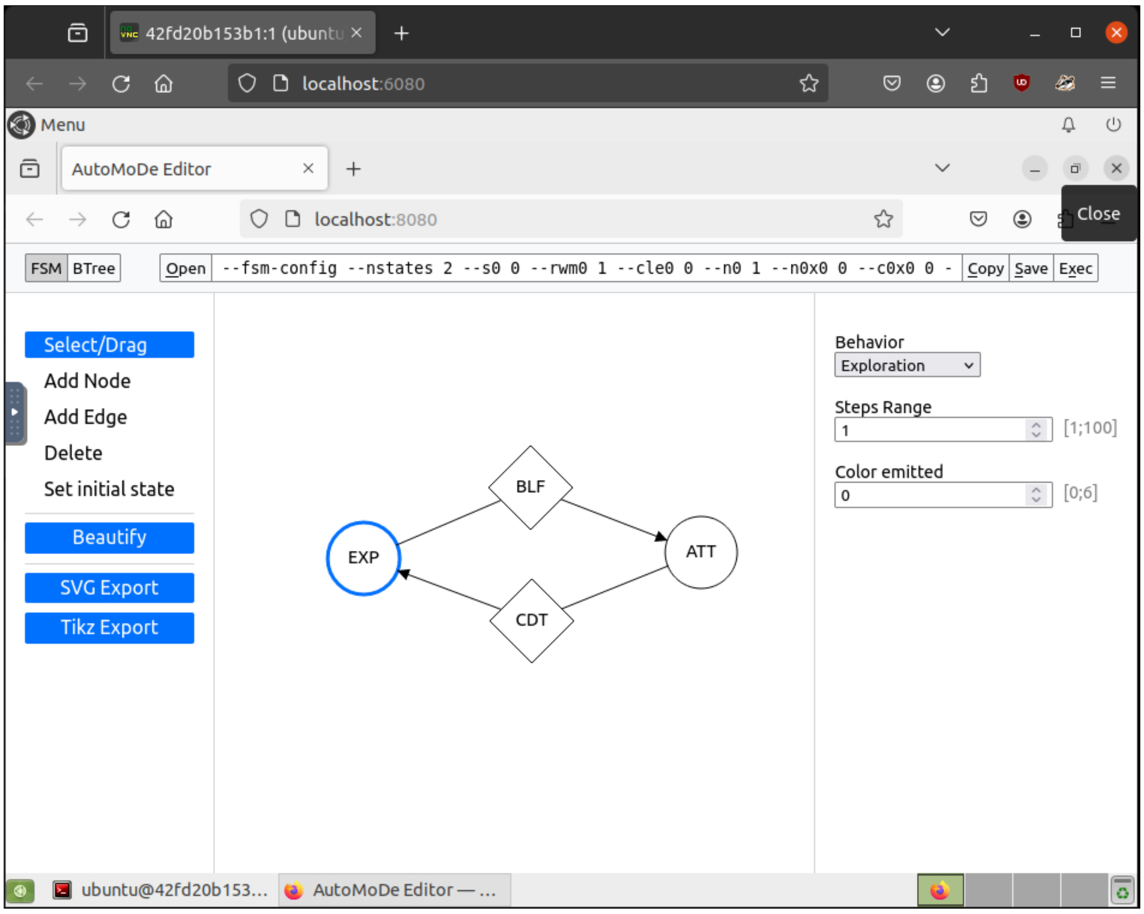
### 1.2 VirtualBox users

A VirtualBox VM with the pre-installed software is provided in the shared Drive of the Summer School. You can download it from Day 5 - Swarm Robotics, and import it in your local installation of VirtualBox.

### 1.3 Native installation

Instructions for a native installation in Ubuntu 20.04 and Ubuntu 22.04 (experimental) are available in the session's repository.

```
https://github.com/dagarzonr/sigsoft-swarms
```



```
--fsm-config --nstates 2 --s0 0 --rwm0 1 --cle0 0 --n0 1 --n0x0 0 --c0x0 0 -
```

Behavior: Exploration

Steps Range: 1 [1;100]

Color emitted: 0 [0;6]

Figure 1: Interface to design the control software for the robots.

## 2 Running the software

After installing the required software, open a new terminal (in the container, VM, or your PC). Enter the work directory and source the relevant environmental variables:

```
cd ~/sigsoft-swarms
source argos3-env.sh
```

Start an experiment by running the following script. Replace ID by the number of the mission you want to test, between 1 and 21—see Section 4.

```
bash start_experiment.sh ID
```

Then, open the following URL in a browser,

```
http://localhost:8080
```

This will spawn an interface that will allow you to create finite state-machines to program the robots and run simulations in ARGoS3—see Figure 1.

The interface allows you to add, remove, and reorganize nodes and edges in your finite state-machine. By clicking in a software module (either a node or an edge) you will have access to the parameters that can be modified in each case. For example, you can select the type of node and its functional parameters. When you are satisfied with your design, click on the `Exec` button to spawn the ARGoS3 simulator and play the simulation to observe the behavior of the robots.

# 3 Designing modular control software for robot swarms

You can design control software in the form of a probabilistic finite-state machine. In this architecture, states (nodes) represent low-level behaviors that the robots execute, and transition conditions (edges) represent events that trigger the change from one behavior to another. Each low-level behavior and transition condition is a parametric software module. You can design finite-state machines that have up to 4 states and up to 4 outgoing transitions from each state. The set of software modules that are available during the experiment is composed of 6 low-level behaviors (nodes) and 7 transition conditions (edges).

## 3.1 Low-level behaviors

Table 1: Set of `TuttiFrutti`'s low-level behaviors.

| **Low-level behavior *** | **Parameters** | **Description** |
|---|---|---|
| EXPLORATION | $\{\tau, \gamma\}$ | movement by random walk |
| STOP | $\{\gamma\}$ | standstill state |
| ATTRACTION | $\{\alpha, \gamma\}$ | physics-based attraction to neighboring robots |
| REPULSION | $\{\alpha, \gamma\}$ | physics-based repulsion from neighboring robots |
| COLOR-FOLLOWING | $\{\delta, \gamma\}$ | steady movement towards robots/objects of color $\delta$ |
| COLOR-ELUSION | $\{\delta, \gamma\}$ | steady movement away from robots/objects of color $\delta$ |

* All low-level behaviors display a color $\gamma \in \{\varnothing, C, M, Y\}$ alongside the action described.

- The parameter `Steps range` ($\tau \in \{1, 100\}$) determines the maximum number of time steps that a robot rotates when it faces an obstacle.
- The parameter `Attraction` ($\alpha \in \{1, 5\}$) is the attraction factor to neighboring robots.
- The parameter `Repulsion` ($\alpha \in \{1, 5\}$) is the repulsion factor from neighboring robots.
- The parameter `Color perceived` ($\delta \in \{1, 6\}$) determines the color to which a robot reacts in a given state.
  - $1 \mapsto$ Red
  - $2 \mapsto$ Green
  - $3 \mapsto$ Blue
  - $4 \mapsto$ Yellow
  - $5 \mapsto$ Magenta
  - $6 \mapsto$ Cyan
- All low-level behaviors enable the robot for displaying a color with its LEDs. The parameter `Color emitted` ($\gamma \in \{0, 6\}$), determines the color that the robot displays.
  - $0 \mapsto$ No color
  - $1 \mapsto$ Red
  - $2 \mapsto$ Green
  - $3 \mapsto$ Blue
  - $4 \mapsto$ Yellow
  - $5 \mapsto$ Magenta
  - $6 \mapsto$ Cyan

Table 2: Set of `TuttiFrutti`'s transition conditions.

| **Transition condition** | **Parameters** | **Description** |
|---|---|---|
| BLACK-FLOOR | $\{\beta\}$ | black floor beneath the robot |
| GRAY-FLOOR | $\{\beta\}$ | gray floor beneath the robot |
| WHITE-FLOOR | $\{\beta\}$ | white floor beneath the robot |
| NEIGHBOR-COUNT | $\{\xi, \eta\}$ | number of neighboring robots greater than $\xi$ |
| INVERTED-NEIGHBOR-COUNT | $\{\xi, \eta\}$ | number of neighboring robots less than $\xi$ |
| FIXED-PROBABILITY | $\{\beta\}$ | transition with a fixed probability |
| COLOR-DETECTION | $\{\delta, \beta\}$ | robots/objects of color $\delta$ perceived |

## 3.2 Transition conditions

- In all transitions, the parameter `Probability` ($\beta \in [0, 1]$) determines the probability of executing a transition when the given condition is fulfilled.
- The parameter $\eta \in \{1, 10\}$ determines the sensitivity at which the transition triggers when the robot is in the presence of $\xi \in \{0, 20\}$ neighboring robots.
- The parameter `Color perceived` ($\delta \in \{1, 6\}$) determines the that triggers the transition.
  - $1 \mapsto$ Red
  - $2 \mapsto$ Green
  - $3 \mapsto$ Blue
  - $4 \mapsto$ Yellow
  - $5 \mapsto$ Magenta
  - $6 \mapsto$ Cyan

# 4 Experimental scenarios

You will experiment with a swarm of twenty e-puck robots that must perform single-criterion missions, or multi-criteria missions composed of two sequential parts [6]. We create missions by combining a set of 6 sub-missions. For each sub-mission, the performance of the swarm is evaluated by an independent objective function: the design criteria.

The robots operate in an octagonal arena of 2.75 $\text{m}^2$ surrounded by RGB blocks [3]—see Figure 2. The robots are randomly positioned at the beginning of each experimental run. The RGB blocks are arranged in walls and each of them can possibly display a different color. The floor of the arena is gray with nine square patches, each measuring 25 cm on each side. One of the patches is white, and the other eight are black. In every mission, RGB blocks adjacent to black patches initially turn green, and afterward, they will randomly switch off with uniform probability. The remaining RGB blocks turn red or blue to inform the robots about the sub-mission to be executed.

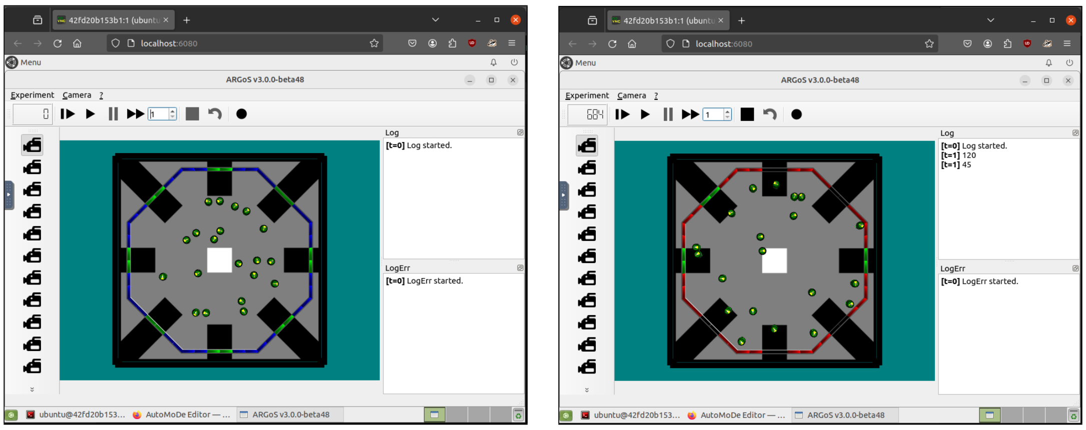
## 4.1 Sub-missions

We consider a set of six sub-missions (S$\{1, \cdots, 6\}$). Each sub-mission is specified by a description of a task to be executed and a corresponding objective function.

### 4.1.1 Sub-mission 1 (S1):

the robots must occupy the black patches whose adjacent RGB blocks display green. The swarm is given 1 point for every 100 cumulative timesteps that the robots spend on each suitable patch. For example, a single robot in a patch will be given one point after 100 timesteps, but 10 robots in a patch will be given 1 point after 10 timesteps. The score of the swarm is the number of points it obtains in the allotted time:

$$f_{\text{S1}} = \sum_{t=1}^{T'} \sum_{i=1}^{H} I_i(t), \tag{1}$$



```
Log
[t=0] Log started.

LogErr
[t=0] LogErr started.
```

```
Log
[t=0] Log started.
[t=1] 120
[t=1] 45

LogErr
[t=0] LogErr started.
```

Figure 2: The experimental arena. The picture shows examples of the two possible states of the arena. On the left, the RGB blocks of the walls display blue and all RGB blocks adjacent to a black patch are switched on, displaying green. On the right, the RGB blocks of the walls display red and some RGB blocks adjacent to a black patch have switched off, displaying no color. The robots are randomly positioned.

which must be maximized. $H$ is the number of patches and $T'$ the time available to the robots to perform the sub-mission. The indicator $I_i(t)$ is defined as:

$$I_i(t) = \begin{cases} 1, \text{ if at time } t \text{ the robots accumulate 100 timesteps in the patch } i; \\ 0, \text{ otherwise.} \end{cases}$$

Sub-mission S1 is inspired by aggregation missions in which the robots must gather at an indicated place [7, 8].

### 4.1.2 Sub-mission 2 (S2):

the robots must iteratively travel from any black patch to the white one. The swarm is given 1 point every time a robot completes a trip. The score of the swarm is the number of points it obtains in the allotted time:

$$f_{\text{S2}} = I_{T'}\,, \tag{2}$$

which must be maximized. $I_{T'}$ is the number of trips executed in the time $T'$ available to the robots to perform the sub-mission. Sub-mission S2 is inspired by foraging missions in which the robots must travel between two locations: a food source and a nest [3, 7].

### 4.1.3 Sub-mission 3 (S3):

the robots must occupy the black patches adjacent to RGB blocks that are switched off. The swarm is given one point if at least two robots spend 50 timesteps in the corresponding black patch. The count of timesteps starts as soon as the two robots step into the patch, and it will continue as long as they both remain on it. The count is not affected if more than two robots occupy the patch. The score of the swarm is the number of points it obtains in the allotted time:

$$f_{\text{S3}} = \sum_{t=1}^{T'} \sum_{i=1}^{H} I_i(t), \tag{3}$$

which must be maximized. $H$ represents the number of patches and $T'$ the time available to the robots to perform the sub-mission. The indicator $I_i(t)$ is defined as:

$$I_i(t) = \begin{cases} 1, \text{ if at time } t \text{ two robots accumulate 50 timesteps in the patch } i;; \\ 0, \text{ otherwise.} \end{cases}$$

Sub-mission S3 is inspired by strictly cooperative missions in which the robots must jointly perform a single task [9, 10].

#### 4.1.4 Sub-mission 4 (S4):

the robots must iteratively enter and leave the white patch. The swarm is awarded 1 point every time a robot performs these two actions. The score of the swarm is the number of points it obtains in the allotted time:

$$f_{\mathrm{S4}} = I_{T'} \,, \tag{4}$$

which must be maximized. $I_{T'}$ is the number of times a robot entered and left the white patch in the time $T'$ that is available to the robots to perform the sub-mission. Also sub-mission S4 is inspired by foraging missions, as sub-mission S2, but here, the robots start and end at a single location.

#### 4.1.5 Sub-mission 5 (S5):

the robots must disperse and cover the arena. We consider the coverage to be the most effective when the minimum distance between any two pair of robots is maximized. The score of the swarm is the cumulative sum of the minimum inter-robot distance, over time:

$$f_{\mathrm{S5}} = \sum_{t=1}^{T'} \min\big(d_{ij}(t)\big) \,, \tag{5}$$

which must be maximized. Here, $d_{ij}$ is the minimum distance between any pair of robots $(i, j)$ at time $t$, and $T'$ is the time available to the robots to perform the sub-mission. Sub-mission S5 is inspired by dispersion and coverage missions in which the robots must maintain a fixed inter-robot distance to achieve a specific spatial distribution [8, 11].

#### 4.1.6 Sub-mission 6 (S6):

the robots must remain within a 25 cm distance from the walls of the arena, without entering the black patches. The score of the swarm is the aggregate time that the robots spend in the suitable areas:

$$f_{\mathrm{S6}} = \sum_{t=1}^{T'} \sum_{i=1}^{N} I_i(t) \,, \tag{6}$$

which must be maximized. $N$ is the number of robots and $T'$ is the time available to the robots to perform the sub-mission. The indicator $I_i(t)$ is defined as:

$$I_i(t) = \begin{cases} 1, \text{ if the robot } i \text{ is in gray floor and in a 25 cm distance from a wall at time } t; \\ 0, \text{ otherwise.} \end{cases}$$

Sub-mission S6 is inspired by missions in which the robots must display a specific spatial distribution, like sub-mission S5 [8, 11]. However, the robots here must maintain a specific distance from an element in their environment, irrespective of the distance to their peers.

### 4.2 Single-criteria missions

We define the set $\mathrm{M}_s$ of single-criteria missions ($m_{\mathrm{S}p.\mathrm{S}p}$) by evaluating a single sub-mission in both parts of the mission, where $p = q$. This results in 6 missions—see Table 3. The time $T$ available to the robots to execute a mission is 120 s. Accordingly, the swarm's performance in a mission is assessed for the whole time regarding a single performance metric. A single score is returned after each experimental run. We expect the swarm to be able to perform a mission $m_{\mathrm{S}p.\mathrm{S}q}$ regardless of the order of S$p$ and S$q$.

Table 3: Set of single-criteria missions ($M_s$). The missions are execution of each of the six sub-missions ($S\{1, \cdots, 6\}$). The colors blue and red characterize the sub-missions S$p$ (🟦) and S$q$ (🟥). In a mission ($m_{Sp.Sq}$), the sub-missions S$p$ and S$q$ represent the same sub-mission ($p = q$), but the order of the sequence and colors (S$p$ 🟦 $\rightleftarrows$ 🟥 S$q$) is randomly defined in every experimental run. S$p$ and S$q$ are executed during an equivalent period of time. The execution of a mission $m_{Sp.Sq}$ returns the a single score for the swarm with respect to the corresponding sub-mission, regardless the order of the sequence.

| **ID** | **Mission** | **Combination of sub-missions** | | |
|---|---|---|---|---|
| 1 | $m_{S1.S1}$ | S1 | 🟦 $\rightleftarrows$ 🟥 | S1 |
| 2 | $m_{S2.S2}$ | S2 | 🟦 $\rightleftarrows$ 🟥 | S2 |
| 3 | $m_{S3.S3}$ | S3 | 🟦 $\rightleftarrows$ 🟥 | S3 |
| 4 | $m_{S4.S4}$ | S4 | 🟦 $\rightleftarrows$ 🟥 | S4 |
| 5 | $m_{S5.S5}$ | S5 | 🟦 $\rightleftarrows$ 🟥 | S5 |
| 6 | $m_{S6.S6}$ | S6 | 🟦 $\rightleftarrows$ 🟥 | S6 |

### 4.3 Multi-criteria missions

We define the set M of multi-criteria missions ($m_{Sp.Sq}$) by pairing sub-missions (S$p$, S$q$) in fifteen combinations—see Table 4. Unlike the single-criteria missions, in this case $p \neq q$. In all cases, the robots must execute the two sub-missions, one after the other. The time $T$ available to the robots to execute a mission is 120 s. The time $T'$ available to execute each sub-mission is 60 s. Accordingly, the swarm's performance in a mission is assessed for the initial 60 s with regard to one sub-mission and for the remaining 60 s with regard to the other. The two scores are returned after each experimental run. We expect the swarm to be able to perform a mission $m_{Sp.Sq}$ regardless of the order of S$p$ and S$q$.

## 5 To be considered

While conducting the experiments, consider the following questions and reflect on how they influence the design process you use to achieve a satisfactory solution for a mission.

- How do you establish communication between robots and how can the use it to display spatial-organization behaviors?
- How many robots are contributing to score better in the proposed missions?
- How do you know whether the swarm is already operating in a satisfactory manner? How do you know if the swarm has a sufficiently good performance?
- Does the performance of the swarm vary from one run to the other? If so, what can cause such variance?
- Can you think on a strategy to decide on the effort you devote to address each mission?
- What factors influence your design process in the multi-criteria mission? Are there reasons to favor one criteria over the other?

## 6 Contact information

If you require further information, do not hesitate to contact us.

David Garzón Ramos
david.garzonramos@bristol.ac.uk
Research Associate - University of Bristol

Table 4: Set of multi-criteria missions (M). The missions are paired combinations ($m_{\mathrm{S}p.\mathrm{S}q}$) of the six sub-missions (S$\{1,\cdots,6\}$). In each combination, the colors blue and red characterize the sub-missions S$p$ (🟦) and S$q$ (🟥). In a mission ($m_{\mathrm{S}p.\mathrm{S}q}$), the sub-missions S$p$ and S$q$ must be executed in sequence, but the order of the sequence (S$p$ 🟦 $\rightleftarrows$ 🟥 S$q$) is randomly defined in every experimental run. S$p$ and S$q$ are executed during an equivalent period of time. The execution of a mission $m_{\mathrm{S}p.\mathrm{S}q}$ returns the score of the swarm with respect to S$p$ and S$q$, regardless the order of the sequence.

| **ID** | **Mission** | **Combination of sub-missions** | | |
|---|---|---|---|---|
| 7 | $m_{\mathrm{S1.S2}}$ | S1 | 🟦 $\rightleftarrows$ 🟥 | S2 |
| 8 | $m_{\mathrm{S1.S3}}$ | S1 | 🟦 $\rightleftarrows$ 🟥 | S3 |
| 9 | $m_{\mathrm{S1.S4}}$ | S1 | 🟦 $\rightleftarrows$ 🟥 | S4 |
| 10 | $m_{\mathrm{S1.S5}}$ | S1 | 🟦 $\rightleftarrows$ 🟥 | S5 |
| 11 | $m_{\mathrm{S1.S6}}$ | S1 | 🟦 $\rightleftarrows$ 🟥 | S6 |
| 12 | $m_{\mathrm{S2.S3}}$ | S2 | 🟦 $\rightleftarrows$ 🟥 | S3 |
| 13 | $m_{\mathrm{S2.S4}}$ | S2 | 🟦 $\rightleftarrows$ 🟥 | S4 |
| 14 | $m_{\mathrm{S2.S5}}$ | S2 | 🟦 $\rightleftarrows$ 🟥 | S5 |
| 15 | $m_{\mathrm{S2.S6}}$ | S2 | 🟦 $\rightleftarrows$ 🟥 | S6 |
| 16 | $m_{\mathrm{S3.S4}}$ | S3 | 🟦 $\rightleftarrows$ 🟥 | S4 |
| 17 | $m_{\mathrm{S3.S5}}$ | S3 | 🟦 $\rightleftarrows$ 🟥 | S5 |
| 18 | $m_{\mathrm{S3.S6}}$ | S3 | 🟦 $\rightleftarrows$ 🟥 | S6 |
| 19 | $m_{\mathrm{S4.S5}}$ | S4 | 🟦 $\rightleftarrows$ 🟥 | S5 |
| 20 | $m_{\mathrm{S4.S6}}$ | S4 | 🟦 $\rightleftarrows$ 🟥 | S6 |
| 21 | $m_{\mathrm{S5.S6}}$ | S5 | 🟦 $\rightleftarrows$ 🟥 | S6 |

## References

[1] Marco Dorigo, Mauro Birattari, and Manuele Brambilla. Swarm robotics. *Scholarpedia*, 9(1): 1463, 2014. `doi:10.4249/scholarpedia.1463`.

[2] Manuele Brambilla, Eliseo Ferrante, Mauro Birattari, and Marco Dorigo. Swarm robotics: a review from the swarm engineering perspective. *Swarm Intelligence*, 7(1):1–41, 2013. `doi:10.1007/s11721-012-0075-2`.

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

[4] Jonas Kuckling, Ken Hasselmann, Vincent van Pelt, Cédric Kiere, and Mauro Birattari. AutoMoDe Editor: a visualization tool for AutoMoDe. Technical Report TR/IRIDIA/2021-009, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium, 2021.

[5] Carlo Pinciroli, Vito Trianni, Rehan O'Grady, Giovanni Pini, Arne Brutschy, Manuele Brambilla, Nithin Mathews, Eliseo Ferrante, Gianni A. Di Caro, Frederick Ducatelle, Mauro Birattari, Luca Maria Gambardella, and Marco Dorigo. ARGoS: a modular, parallel, multi-engine simulator for multi-robot systems. *Swarm Intelligence*, 6(4):271–295, 2012. `doi:10.1007/s11721-012-0072-5`.

[6] David Garzón Ramos, Federico Pagnozzi, Thomas Stützle, and Mauro Birattari. Automatic design of robot swarms under concurrent design criteria: a study based on Iterated F-Race. *Advanced Intelligent Systems*, page (under review), 2024.

[7] Gianpiero Francesca, Manuele Brambilla, Arne Brutschy, Vito Trianni, and Mauro Birattari. AutoMoDe: a novel approach to the automatic design of control software for robot swarms. *Swarm Intelligence*, 8(2):89–112, 2014. `doi:10.1007/s11721-014-0092-4`.

[8] Gianpiero Francesca, Manuele Brambilla, Arne Brutschy, Lorenzo Garattoni, Roman Miletitch, Gaëtan Podevijn, Andreagiovanni Reina, Touraj Soleymani, Mattia Salvaro, Carlo

Pinciroli, Franco Mascia, Vito Trianni, and Mauro Birattari. AutoMoDe-Chocolate: automatic design of control software for robot swarms. *Swarm Intelligence*, 9(2–3):125–152, 2015. doi:10.1007/s11721-015-0107-9.

[9] Auke Jan Ijspeert, Alcherio Martinoli, Aude Billard, and Luca Maria Gambardella. Collaboration through the exploitation of local interactions in autonomous collective robotics: the stick pulling experiment. *Autonomous Robots*, 11(2):149–171, 2001. doi:10.1023/A:1011227210047.

[10] Vito Trianni and Manuel López-Ibáñez. Advantages of task-specific multi-objective optimisation in evolutionary robotics. *PLOS ONE*, 10(8):e0136406, 2015. doi:10.1371/journal.pone.0136406.

[11] Fernando J. Mendiburu, David Garzón Ramos, Marcos Ricardo A. Morais, Antonio Marcus Nogueira Lima, and Mauro Birattari. AutoMoDe-Mate: automatic off-line design of spatially-organizing behaviors for robot swarms. *Swarm and Evolutionary Computation*, 74:101118, 2022. doi:10.1016/j.swevo.2022.101118.