applied sciences

MDPI

Article

Automatic Design of Collective Behaviors for Robots that Can Display and Perceive Colors

David Garzón Ramos * and Mauro Birattari *

Institut de Recherches Interdisciplinaires et de Développements en Intelligence Artificielle (IRIDIA), Université Libre de Bruxelles, 1050 Brussels, Belgium

* Correspondence: dgarzonr@ulb.ac.be (D.G.R.); mbiro@ulb.ac.be (M.B.); Tel.: +32-02-6502729 (D.G.R.)

Received: 16 June 2020; Accepted: 1 July 2020; Published: 6 July 2020

Abstract: Research in swarm robotics has shown that automatic design is an effective approach to realize robot swarms. In automatic design methods, the collective behavior of a swarm is obtained by automatically configuring and fine-tuning the control software of individual robots. In this paper, we present `TuttiFrutti`: an automatic design method for robot swarms that belongs to AutoMoDe—a family of methods that produce control software by assembling preexisting software modules via optimization. The peculiarity of `TuttiFrutti` is that it designs control software for e-puck robots that can display and perceive colors using their RGB LEDs and omnidirectional camera. Studies with AutoMoDe have been so far restricted by the limited capabilities of the e-pucks. By enabling the use of colors, we significantly enlarge the variety of collective behaviors they can produce. We assess `TuttiFrutti` with swarms of e-pucks that perform missions in which they should react to colored light. Results show that `TuttiFrutti` designs collective behaviors in which the robots identify the colored light displayed in the environment and act accordingly. The control software designed by `TuttiFrutti` endowed the swarms of e-pucks with the ability to use color-based information for handling events, communicating, and navigating.

Keywords: swarm robotics; automatic design; AutoMoDe; evolutionary robotics

1. Introduction

A robot swarm [1,2] is a group of robots that operate autonomously without relying on a leader robot or on external infrastructures. By cooperating, the robots of a swarm can collectively accomplish missions that individual robots could not accomplish alone. The collective behavior of a robot swarm—and hence its ability to accomplish a particular mission—is the result of the interactions that the robots have with the environment and with their peers [3].

Unfortunately, conceiving and implementing a collective behavior for a robot swarm is particularly challenging. Indeed, to obtain a collective behavior, one must conceive and implement the control software of the individual robots. The problem is that no generally applicable method exists to tell what an individual robot should do so that the desired behavior is obtained [4]. Automatic design is a promising approach to address this problem. An automatic design method produces control software via an optimization algorithm that maximizes an appropriate mission-dependent objective function. For a recent literature review on the automatic design of robot swarms, see Francesca et al. [5].

Traditionally, research on the automatic design of robot swarms adopts the neuro-evolutionary approach [6,7]. Design methods based on neuro-evolution produce control software in the form of artificial neural networks. The architecture and parameters of the network are selected by an evolutionary algorithm. As an alternative to neuro-evolution, some modular methods have been proposed [8–14]. In the modular approach, preexisting software modules are combined and tuned by an optimization algorithm. Results show that modular methods are more suitable to produce

communication-based behaviors [14] and are more robust to the so-called reality gap [8,15], that is, the possibly subtle but unavoidable differences between reality and the simulation models used in the design process.

In this paper, we present `TuttiFrutti`: a method for the automatic design of swarms of e-pucks (extended with an omnidirectional vision turret [16]) that can display and perceive colors. `TuttiFrutti` designs control software for the individual robots in the swarm by selecting, tuning, and assembling preexisting software modules into probabilistic finite state machines. `TuttiFrutti` is an instance of AutoMoDe [8]—a family of modular methods for the realization of robot swarms. `TuttiFrutti` differentiates from previous instances of AutoMoDe by enabling the production of control software that operates with information expressed in the form of colors. More precisely, `TuttiFrutti` is intended to solve classes of missions in which robots shall act according to colors displayed by objects in their environment and/or their peers. With `TuttiFrutti`, we significantly enlarge the variety of collective behaviors that can be obtained by AutoMoDe. The study we present in this paper is framed within the tenets of the automatic off-line design of robot swarms, as recently defined by Birattari et al. [17]: (i) `TuttiFrutti` is not intended to solve a specific design problem but rather a class thereof, without the need to undergo any problem-specific modification or adjustment; (ii) once a design problem is specified, human intervention is not provided for in any phase of the design process.

In our research, we address the following questions: Is `TuttiFrutti` capable of deciding whether a color displayed in the environment provides information useful to accomplish a mission? Can `TuttiFrutti` produce collective behaviors that exhibit color-based communication between robots? Do the extended capabilities of the e-puck increase the difficulty of automatically designing control software for the robot swarm? How could these new resources be used to create more complex missions?

We consider a model of the e-puck that can use its RGB LEDs for emitting color signals, and its omnidirectional vision turret [16] for detecting robots or other objects that display colors in the environment. We conduct our study to demonstrate that e-pucks that display and perceive colors enable the automatic design of collective behaviors with event-handling, communication and navigation properties. As a proof of concept, we assess `TuttiFrutti` in three missions in which colors displayed in the environment play a different role: STOP, AGGREGATION, and FORAGING. In STOP, the robots must stop moving as soon as a color signal appears in the environment. In AGGREGATION, the robots must aggregate in a region where a specific color is displayed. In FORAGING, the robots must forage in an environment that has two sources of items—the sources differ in the profit they provide and in the color displayed at their location. We report a statistical analysis of results obtained with realistic computer simulations and with a swarm of e-puck robots.

Alongside the results of `TuttiFrutti`, we report the results obtained by `EvoColor`—a design method based on neuro-evolution. `EvoColor` is a straightforward implementation of the neuro-evolutionary approach that, likewise `TuttiFrutti`, produces control software for swarms of e-pucks that can display and perceive colors. We report these results as a reference for appraising the complexity of the missions considered in our study. In the absence of a well established state-of-the-art off-line design method for the missions proposed here, we consider `EvoColor` as a reasonably appropriate yardstick against which we can assess the performance of `TuttiFrutti`. A thorough comparison of `TuttiFrutti` against any possible declination of the neuro-evolutionary approach is well beyond the scope of this paper.

The paper is structured as follows: Section 2 discusses previous related work; Section 3 introduces `TuttiFrutti`; Section 4 describes the experimental set-up; Section 5 presents the results; and Section 6 concludes the paper and highlights future work.

2. Related Work

In this section, we first introduce studies in which the robots of a swarm have the capabilities of displaying and perceiving colors. After, we revise related works on automatic design of robot swarms. Finally, we compare `TuttiFrutti` with the other existing instances of AutoMoDe.

Robots that can display or perceive colors have been largely used to demonstrate collective behaviors in swarm robotics. The literature on robot swarms that use visual information is extensive and it is not our intention to provide an exhaustive review. We exclude from our discussion any system in which robots only perceive visual information but do not display it—such as [18–22]. Instead, we focus on studies in which the robots both display and perceive colors to achieve collective behaviors [23–47].

Designers of robot swarms commonly use color lights to represent specific information that robots in the swarm must identify, process and/or transmit—the nature of the information varies from one study to another and is used ad hoc to obtain a particular behavior. For example, Nouyan et al. [35] designed a swarm that connects locations by establishing a chain of robots that act as waypoints for their peers. They conducted two experiments in which robots use colors differently: in the first experiment, robots repeat a pattern of 3 colors along the chain to indicate the sense in which the peers should move; in the second one, robots use colors to inform their peers about a location of interest. Mathews et al. [25] designed a swarm in which robots self-organize in mergeable structures. In their experiments, robots react to colored objects in their environment and display color signals that indicate their location. Garattoni and Birattari [31] designed a robot swarm that autonomously identifies and perform sequences of tasks. In their experiments, robots emit color signals to coordinate their collective action and associate each task in a sequence with objects that display a particular color. In a more general sense, one can find a similar approach in swarms that exhibit self-assembly and morphogenesis [23–26], collective fault detection [25,27], collective exploration [28–31], collective transport [28,30], coordinated motion [25,26,32], human-swarm interaction [33,34], chain formation [28,31,35], group size regulation [36], task allocation [31,37–40], object clustering [39,41], and foraging [42]—according to the taxonomy proposed by Brambilla et al. [4]. In these studies [23–42], designers manually established ad hoc relationships between the colors that a robot can perceive and the corresponding behavior that a robot must adopt when it perceives them. The research question we address in the present paper is whether automatic design methods [5,17] can establish similar relationships.

It is our contention that classes of missions that require the robots to communicate and react to color-based information are an appropriate benchmark to assess automatic design methods. First, the capability of displaying and perceiving colors is platform-independent and generalizes across different design methods—robot platforms used in swarm robotics often include LEDs and cameras [48]. Second, colors facilitate the conception and realization of complex missions—colored environments can be created in various manners [40,41,49–51]. Finally, colors simplify the visualization and monitoring of robot swarms [48]—a property relevant to the human understandability of collective behaviors, an open research area in swarm robotics [52]. Yet, no existing method for the automatic design of robot swarms targets classes of missions that require the robots to operate with color-based information. Specifically, we refer to methods that can be framed within the tenets of the automatic off-line design of robot swarms [17].

Few related studies have been conducted following the neuro-evolutionary approach [43–47]. Floreano et al. [43] evolved communication behaviors for a swarm performing a foraging mission. Ampatzis et al. [44] evolved self-assembly behaviors with a team of two robots. Sperati et al. [45,46] evolved behaviors for coordinated motion with a group of three robots, and later, evolved a dynamic chain of robots that perform a foraging-like mission. Trianni and López-Ibañez [47] used multi-objective optimization to evolve flocking and a two-robot collaborative behavior within a robot swarm. In the studies mentioned above, the methods under analysis have not been tested for their ability to generate control software autonomously. As a consequence, these studies belong in semi-automatic design

rather than in automatic design [17]. Indeed, in these studies, researchers either focused a single mission [43,44,46] or modify the design methods and/or robot platform when they applied the methods under analysis to more than one [45,47]. In addition, we argue that mission-specific bias was manually introduced in the way the robots display and perceive colors: robots display colors that are manually defined at the beginning of the experiment [44]; robots display color-based information that is encoded by the researchers [46,47]; and the perception capabilities of the robots are adjusted to ease the emergence of a specific behavior [45,46]. We contend that these studies do not expose the full potential of using color-based information in the automatic design of collective behaviors. As a matter of fact, previous works were limited to produce control software for robots than can display and perceive a single [43–45] or at most two simultaneous colors [46,47].

Our research belongs in the family of modular design methods known as AutoMoDe [8]. In the rest of the section, we restrict our attention to this family. Francesca and Birattari [5] discussed how the capabilities of robot platforms limit the variety of collective behaviors that automatic design methods can produce. Methods conceived for e-puck robots with equal capabilities—such as `Vanilla` [8], `Chocolate` [9], `Maple` [10], `Waffle` [11], `Coconut` [12] and `IcePop` [13]—are restricted to address the same class of missions: robots in the swarm must position themselves regarding their peers [9,15] or few static environmental features [8–13,15,53], and they can only use a single global reference for navigation [8–13,15,53]. In contrast, Hasselmann et al. [14] obtained a larger variety of collective behaviors by considering an extended set of capabilities with respect to those considered in `Chocolate`. They introduced `Gianduja`—a method for the automatic design of swarms of e-pucks that can selectively broadcast binary messages. Hasselmann et al. showed that by broadcasting and reacting to messages, the robots can perform missions that `Chocolate` cannot address—for example, missions that require event-handling collective behaviors.

The approach we follow in our research is similar to the one of Hasselmann et al. [14]. We conceived `TuttiFrutti` as an instance of AutoMoDe that designs control software for e-pucks with the extended capability of communicating by displaying and perceiving colors. As we will see in Section 3, `TuttiFrutti` can address missions that require the robots to act according to color-based information—a class of missions that existing instances of AutoMoDe can not address.

3. AutoMoDe-`TuttiFrutti`

`TuttiFrutti` is a modular automatic off-line design method; it produces control software for swarms of e-pucks that can display and perceive colors. More precisely, `TuttiFrutti` is an instance of AutoMoDe specialized in the design of robot swarms that act according to color-based information. The variety of collective behaviors produced by previous instances of AutoMoDe have been so far restricted by the limited capabilities of the robots—see Section 2. We conceive `TuttiFrutti` to overcome this restriction from a twofold perspective: on the one hand, e-pucks that display and perceive colors could enable the design of robot swarms in which the individuals exhibit color-based communication; on the other hand, these swarms could perform missions in complex and time-varying environments. By introducing communication capabilities and missions with complex environments, we meant to enlarge significantly the variety of collective behaviors designed with AutoMoDe.

Three fundamental components characterize `TuttiFrutti`: the robot platform, the set of preexisting software modules, and the optimization process that produces the control software. In the following sub-sections we describe each of these components, their relationship, and how they differentiate from other instances of AutoMoDe.

3.1. Robot Platform

`TuttiFrutti` produces control software for an extended version of the e-puck [54,55]—see Figure 1. The e-puck is a two wheeled, small, educational robot often used in swarm robotics research [8–11,14,45,46]. We consider a model of the e-puck endowed with a set of sensors and actuators defined by the reference model RM 3—see Table 1. We adopt the concept of *reference*

model [5] to formally characterize the platform for which `TuttiFrutti` can produce control software. RM 3 represents the capabilities of the robot both in real and simulated environments.

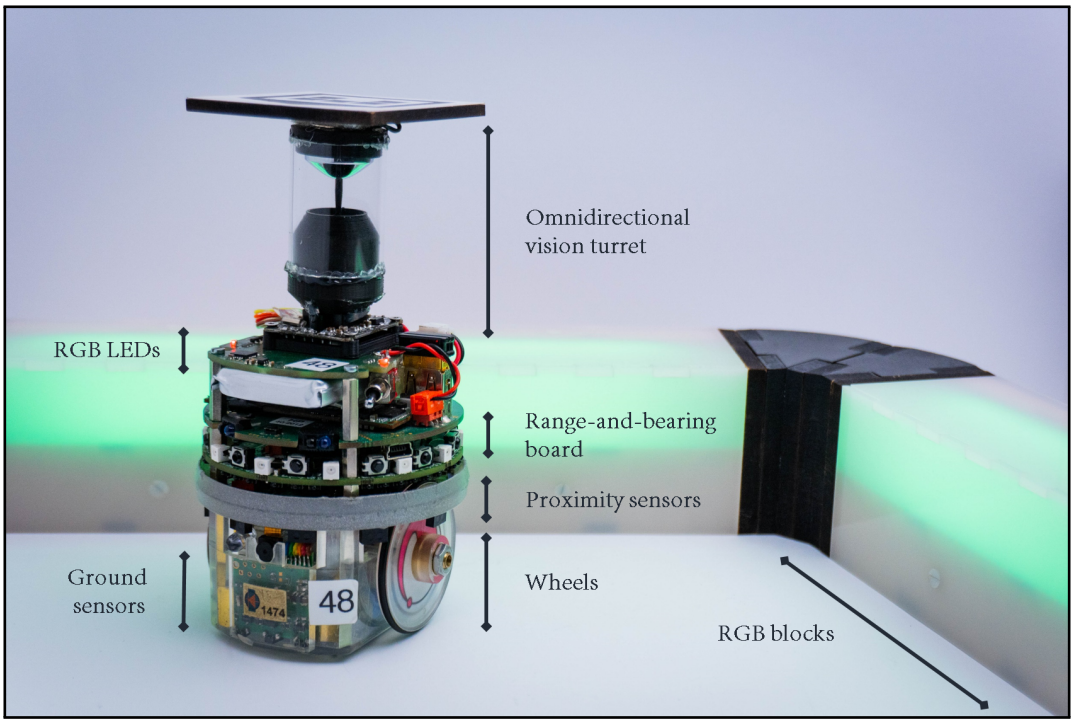

Figure 1. Extended version of the e-puck. The picture indicates the set of sensors and actuators defined by RM 3. Alongside, we show the RGB blocks that we use in our experiments with `TuttiFrutti`.

Table 1. Reference model RM 3. Novelties with respect to RM 1.1 are highlighted in gray. They concern the capabilities of displaying and perceiving colors. Robots can perceive: red (R); green (G); blue (B); cyan (C); magenta (M); and yellow (Y). Robots can display no color $(\varnothing)$; cyan (C); magenta (M); and yellow (Y). V_c is calculated likewise V_n—for each perceived color, the positions of color signals are aggregated into a unique attraction vector.

Input	**Value**	**Description**
$prox_{i\in\{1,\ldots,8\}}$	$[0,1]$	reading of proximity sensor i
$gnd_{j\in\{1,\ldots,3\}}$	$\{black, gray, white\}$	reading of ground sensor j
n	$\{0,\ldots,20\}$	number of neighboring robots detected
V_n	$([0.5,20];[0,2]\,\pi\,\text{rad})$	their relative aggregate position
$cam_{c\in\{R,G,B,C,M,Y\}}$	$\{yes, no\}$	colors perceived
$V_{c\in\{R,G,B,C,M,Y\}}$	$(1.0;[0,2]\,\pi\,\text{rad})$	their relative aggregate direction
Output	**Value**	**Description**
$v_{k\in\{l,r\}}$	$[-0.12,0.12]\,\text{m/s}$	target linear wheel velocity
LEDs	$\{\varnothing, C, M, Y\}$	color displayed by the LEDs

Period of the control cycle: 0.1 s.

The sensors and actuators available to the e-puck are proximity and ground sensors, a range-and-bearing board [56], an omnidirectional vision turret [16], right and left wheels, and RGB LEDs. The e-puck can detect nearby obstacles by its eight proximity sensors $(prox_i)$ distributed around its chassis. Three infrared ground sensors (gnd_j) allow the e-puck to differentiate between black, gray and white floor. By means of its range-and-bearing board, the e-puck knows the number of neighboring peers (n) in a range of 0.5 m. A vector (V_n) represents an attraction force to the neighboring e-pucks—to which the robot is subject—following the framework of virtual physics [57].

The omnidirectional vision turret allows the e-puck to perceive red, blue, green, cyan, magenta and yellow lights (cam_c) in a 360° field of view and within a range of about 0.5 m. For each color perceived, a unit vector (V_c) is associated, which represents a steady attraction to robots or objects that display the color. Finally, the control software of the robot can adjust independently the velocity of each wheel (v_k) between -0.12 and 0.12 m/s, and, using the three RGB LEDs placed on the top of the e-puck, can display cyan, magenta or yellow.

RM 3 is the first reference model adopted in the definition of a design method of the AutoMoDe family that includes the omnidirectional vision turret and the RGB LEDs of the e-puck platform. `Vanilla`, `Chocolate` and `Maple` are based on the simpler RM 1.1 [58]—the differences between RM 3 and RM 1.1 are highlighted in Table 1. `Gianduja` introduced the reference model RM 2 [58], associated to e-pucks that can exchange binary messages using their range-and-bearing board. An important difference between `TuttiFrutti` and other instances of AutoMoDe is that in RM 3 we removed the capability of the e-puck of measuring ambiance light. Although present in RM 1.1 and RM 2, this capability is incompatible with the RGB LEDs we added in RM 3.

3.2. Set of Preexisting Modules

The major characteristic of AutoMoDe is that it produces control software by assembling preexisting software modules. In `TuttiFrutti`, the modules are combined into probabilistic finite state machines—as in `Vanilla`, `Chocolate`, and `Gianduja` [8,9,14]. We conceived a set of modules that comprises six low-level behaviors—actions that a robot can take, and seven transition conditions—situations that trigger the change from one low-lever behavior to another. The set of modules of `TuttiFrutti` adapts and extends the modules originally conceived for `Vanilla`. We designed the modules to operate with RM 3 and they provide the e-puck different means to interact with robots and objects that display colors. Table 2 lists the low-level behaviors and transition conditions of `TuttiFrutti`. We further describe the modules in the following.

Table 2. Set of `TuttiFrutti`'s modules. Novelties with respect to `Vanilla` are highlighted in gray. They concern to the capability of acting upon perceived colors. The modules operate according to RM 3, see Table 1.

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 δ
COLOR-ELUSION	$\{\delta, \gamma\}$	steady movement away from robots/objects of color δ
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 ξ
INVERTED-NEIGHBOR-COUNT	$\{\xi, \eta\}$	number of neighboring robots lower than ξ
FIXED-PROBABILITY	$\{\beta\}$	transition with a fixed probability
COLOR-DETECTION	$\{\delta, \beta\}$	robots/objects of color δ perceived

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

3.2.1. Low-Level Behaviors

In EXPLORATION, the robot moves straight until it detects an obstacle in front $(prox_i)$. Then, it rotates for a number of control cycles determined by the integer parameter τ, in a range of

$\tau \in \{0, \ldots, 100\}$. STOP maintains the robot standing still. In ATTRACTION and REPULSION, the robot moves closer (V_d) or farther from $(-V_d)$ neighboring peers, respectively. In both cases, the velocity of the robot is a function of the number of robots detected (n) and the parameter $\alpha \in [0, 5]$. If the robot does not detect other robots, it moves straight. COLOR-FOLLOWING and COLOR-ELUSION move the robot with constant velocity towards (V_c) or away $(-V_c)$ from robots or objects displaying specific colors (cam_c). The parameter $\delta \in \{R, G, B, C, M, Y\}$ determines the color to which the behaviors react. Robots can display the colors $\delta \in \{C, M, Y\}$, and other objects that might populate the environment can display the colors $\delta \in \{R, G, B\}$. If the robot does not perceive the color determined by δ, it moves straight. ATTRACTION, REPULSION, COLOR-DETECTION and COLOR-ELUSION incorporate a physics-based obstacle avoidance [59]. In all the low-level behaviors, the parameter $\gamma \in \{\varnothing, C, M, Y\}$ determines the color displayed by the RGB LEDs of the robot. The parameters τ, α, δ, and γ are tuned by the automatic design process.

3.2.2. Transition Conditions

BLACK-FLOOR, GRAY-FLOOR and WHITE-FLOOR trigger a transition when the robot steps on a portion of the floor (gnd_j) that is, respectively, black, gray or white. The parameter $\beta \in [0, 1]$ determines the probability of transitioning. NEIGHBOR-COUNT and INVERTED-NEIGHBOR-COUNT are transition conditions that consider the number of neighboring robots (n). NEIGHBOR-COUNT triggers a transition with a probability $z(n) \in [0, 1]$, with $z(n) = \frac{1}{1+e^{\eta(\xi - n)}}$. Conversely, INVERTED-NEIGHBOR-COUNT triggers a transition with a probability of $1 - z(n)$. The parameter $\xi \in \{0, \ldots, 10\}$ determines the inflection point of the probability function $z(n)$, the parameter $\eta \in [0, 20]$ determines its steepness. FIXED-PROBABILITY triggers a transition with a fixed probability determined by $\beta \in [0, 1]$. COLOR-DETECTION is based on the colors perceived by the robot (cam_c). The parameter $\delta \in \{R, G, B, C, M, Y\}$ defines the color that triggers a transition with a probability $\beta \in [0, 1]$. Robots in the swarm can display the colors $\delta \in \{C, M, Y\}$, and other objects that might populate the environment can display the colors $\delta \in \{R, G, B\}$. The parameters β, ξ, η, and δ are tuned by the automatic design process.

EXPLORATION, STOP, ATTRACTION and REPULSION are modified versions of the original low-level behaviors of `Vanilla`. In `TuttiFrutti`, we add the ability to control the color displayed by the LEDs. All the transition conditions, with exception of COLOR-DETECTION, are implementations of the original modules of `Vanilla`. COLOR-FOLLOWING, COLOR-ELUSION, and COLOR-DETECTION are modules we introduce here for the first time.

3.3. Design of Control Software

`TuttiFrutti` produces control software following the automatic design process proposed in `Chocolate` [9] and further studied in `Gianduja` [14]. The design of the control software is translated into an optimization problem—an optimization algorithm selects an appropriate combination of modules and parameters that, when uploaded to each robot, lead the swarm to exhibit a specific collective behavior. The collective behavior results then from an optimization process that maximizes the performance of the swarm, measured by an appropriate mission-specific performance measure. In `TuttiFrutti`, the architecture of the control software is a probabilistic finite state machine with a maximum of four states—each of which is a low-level behavior, and a maximum of four outgoing edges—to each of which a transition condition is associated. Edges always originate and end in different states—self-transitions are not allowed. The modules included in the finite state machine and the values of their parameters are selected off-line—that is, before the swarm is deployed on its target environment. To that purpose, `TuttiFrutti` uses Iterated F-race [60]—a multipurpose optimization method based on F-race [61]—to search the design space for effective control software configurations. The performance of the configurations is estimated through simulations performed in ARGoS3 [62], version beta 48, together with the argos3-epuck library [55]. The duration of the optimization process is determined by an *a priori* defined budget of simulations. Once the budget is exhausted, the design

process terminates and Iterated F-race returns the best configuration found. This configuration is then uploaded to the robots and assessed in the target environment.

4. Experimental Set-Up

In this section, we describe our experiments in the design of collective behaviors for robots that can display and perceive colors. The study evaluates the capabilities of `TuttiFrutti` to address a class of missions in which colors displayed by objects in the environment provide relevant information to the robots. In the following, we first introduce the missions we consider in our study. Then, we present `EvoColor`—a baseline design method that serves as a reference to appraise the complexity of the missions. Finally, at the end of the section we describe the protocol we follow to assess `TuttiFrutti`.

4.1. Missions

We conceived three missions in which robot swarms operate in arenas surrounded by modular RGB blocks: STOP, AGGREGATION, and FORAGING. STOP and AGGREGATION are adaptations we make from the equivalent missions proposed by Hasselmann et al. to study `Gianduja` [14]. FORAGING is an abstraction of a foraging task, in a *best-of-n* fashion [63]—similar to the experiment of Valentini et al. [64]. The performance of the swarm is evaluated according to a mission-dependent objective function. We selected these missions because we conjecture that, to successfully perform them, the robots need to identify, process and/or transmit color-based information. In all missions, the time available to the robots is $T = 120\,\text{s}$. The RGB blocks are arranged in walls that display colors on a per-mission basis—each RGB block is 0.25 m length and can display the colors red, green and blue $\{R, G, B\}$. In the context of these missions, when we reference to colored walls we imply that the RGB blocks arranged in the wall display the named color—for example, *"the green wall"* stands for a wall in which the RGB blocks composing it display the color green. In the following, we describe the scenario, the objective function, and the role of the colors for each mission. Figure 2 shows the arenas for the three missions.

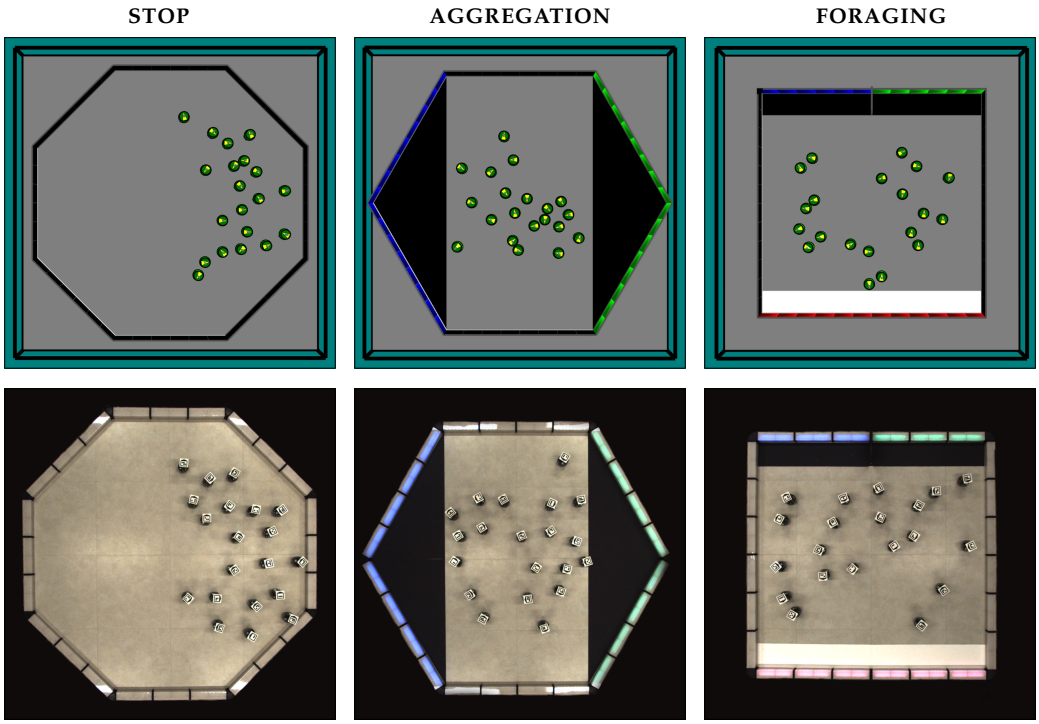

Figure 2. Set-up of the simulated (top) and real arena (bottom) for the missions STOP (left), AGGREGATION (center), and FORAGING (right). The images show an example of the initial position of the robots.

4.1.1. STOP

The robots must move until one of the walls that surrounds the arena emits a stop signal by turning green. Once the wall turns green, all the robots in the swarm must stop moving as soon as possible. The swarm operates in an octagonal arena of 2.75 m^2 and gray floor. The wall that emits the stop signal is selected randomly. At the beginning of each run, the robots are positioned in the right side of the arena. Figure 2 (left) shows the arena for STOP.

The performance of the swarm (C_s) is measured by the objective function described by Equation (1); the lower the better.

$$C_s = \sum_{t=1}^{\bar{t}} \sum_{i=1}^{N} \bar{I}_i(t) + \sum_{t=\bar{t}+1}^{T} \sum_{i=1}^{N} I_i(t); \tag{1}$$

$$I_i(t) = \begin{cases} 1, \text{ if robot } i \text{ is moving at time } t; \\ 0, \text{ otherwise;} \end{cases} \qquad \bar{I}_i(t) = 1 - I_i(t).$$

C_s measures the amount of time during which the robots do not perform the intended behavior—before and after the stop signal. N and T represent respectively the number of robots and the duration of the mission. $\bar{t}$ indicates the time at which the stop signal is displayed. The time $\bar{t}$ is uniformly sampled between $[40, 60]$ s. We expect that `TuttiFrutti` produces collective behaviors with event-handling capabilities that allow the swarm to react when the stop signal appears.

4.1.2. AGGREGATION

The robots must aggregate in the left black region of the arena as soon as possible. The swarm operates in a hexagonal arena of about 2.60 m^2 and gray floor. Triangular black regions of about 0.45 m^2 are located at the left and right side of the arena. The walls lining the left black region are blue and those lining the right black region are green—the colors do not change during the mission. Each black region is characterized by the color of the walls that lines it. That is, the *blue zone* refers to the black region lined by blue walls and the *green zone* refers to the black region lined by green walls. At the beginning of each run, the robots are randomly positioned in the center of the arena—between the black regions. Figure 2 (center) shows the arena for AGGREGATION.

The performance of the swarm (C_a) is measured by the objective function described by Equation (2); the lower the better.

$$C_a = \sum_{t=1}^{T} \sum_{i=1}^{N} I_i(t) \tag{2}$$

$$I_i(t) = \begin{cases} 1, \text{ if robot } i \text{ is not in the aggregation area at time } t; \\ 0, \text{ otherwise.} \end{cases}$$

C_a indicates the time that the robots spend outside of the blue zone. N and T represent the number of robots and the duration of the mission, respectively. We expect that `TuttiFrutti` produces collective behaviors in which the swarm uses the blue walls as a reference to navigate and aggregate in the blue zone.

4.1.3. FORAGING

The robots must select and forage from the most profitable of two sources of items. The swarm operates in a squared arena of 2.25 m^2 and gray floor. A rectangular white region of about 0.23 m^2 is located at the bottom of the arena and represents the nest of the swarm. A rectangular black region of 0.23 m^2 is located at the top of the arena and represents the two sources of items—the sources are separated by a short wall segment that does not display any color. This wall segment divides the black region in half. We account that an item is transported and successfully delivered when a robot travels

from any of the sources to the nest. The walls lining the nest are red, the walls lining the left source are blue, and the walls lining the right source are green—the colors do not change during the mission. We consider then two types of sources of items: the *blue source*—the black region lined by blue walls; and the *green source*—the black region lined by green walls. At the beginning of each run, the robots are randomly positioned in the center of the arena—between the white and black areas. Figure 2 (right) shows the arena for FORAGING.

The performance of the swarm (C_f) is measured by the objective function described by Equation (3); the higher the better.

$$C_f = (\kappa) I_b + (-\kappa) I_g; \tag{3}$$
$$\kappa = 1.$$

C_f indicates the aggregate profit of the total of items collected from the two sources. I_b corresponds to the number of items collected from the blue source, and I_g corresponds number of items collected from the green one. We added the factor κ to balance the profit of the items available in each source. In our study $\kappa = 1$. Items from the blue source account for a profit of $+1$ and items from the green source account for a penalization of -1. We expect that `TuttiFrutti` produces collective behaviors in which swarms use the blue walls as a reference to navigate towards the blue source, the green walls for avoiding the green source, and the red walls to navigate towards the nest.

4.2. Baseline Method: `EvoColor`

No standard design method exists to address the class of missions we consider in this study. Little related work exists—see Section 2—and refers only to mission-specific methods that follow the neuro-evolutionary approach. Indeed, as no extensive comparison has ever been performed between neuro-evolutionary methods across multiple missions, a state of the art in neuro-evolutionary robotics has not been identified, yet. Together with the results obtained with `TuttiFrutti`, in the following we will present also those obtained by `EvoColor`—a method based on neuro-evolution for the automatic design of swarms of e-pucks that can display and perceive colors. The results we will present should not therefore be considered as a comparison between `TuttiFrutti` and the state of the art in neuro-evolutionary robotics. In this context, our results should rather be considered as a comparison between `TuttiFrutti` and a reasonable instance of the neuro-evolutionary approach.

`EvoColor` is an adaptation of `EvoStick` [65]—a standard neuro-evolutionary method previously used as a yardstick in studies on the automatic design of robot swarms [8,10,15,53]. To the best of our knowledge, `EvoStick` is the only neuro-evolutionary method that has been tested via simulations and robot experiments on multiple missions without undergoing any per-mission modification. `EvoColor` produces control software for swarms of e-pucks that operate with RM 3—see Section 3.1. The control software has the form of a fully connected feed-forward artificial neural network with 41 input nodes (in), 8 output nodes (out) and no hidden layers. In this topology, the input nodes and output nodes are directly connected by synaptic connections $(conn)$ with weights (ω) in a range of $[-5, 5]$. The activation of each output node is determined by the weighted sum of all inputs nodes filtered through a standard logistic function. `EvoColor` selects appropriate synaptic weights using an evolutionary process based on elitism and mutation. Just as in `TuttiFrutti`, the evolutionary process is conducted through simulations performed in ARGoS3, version beta 48, together with the argos3-epuck library. The evolution ends when an *a priori* defined budget of simulations is exhausted. Table 3 summarizes the topology of the neural network, the novelties with respect to `EvoStick` and the parameters used in the evolutionary process.

Table 3. Network topology and parameters of the evolutionary process in `EvoColor`. Novelties with respect to `EvoStick` are highlighted in gray. They concern the capability of displaying and perceiving colors. The neural network operates according to RM 3, see Table 1.

Input Node	**Description**
$in_{a\in\{1,\ldots,8\}}$	readings of proximity sensors $prox_{i\in\{1,\ldots,8\}}$
$in_{a\in\{9,\ldots,11\}}$	readings of ground sensors $gnd_{j\in\{1,\ldots,3\}}$
$in_{a\in\{12\}}$	value of the density function $z'(n)$
$in_{a\in\{13,\ldots,16\}}$	scalar projections of V_n
$in_{a\in\{17,\ldots,40\}}$	scalar projections of $V_{c\in\{R,G,B,C,M,Y\}}$
$in_{a\in\{41\}}$	bias input
Output Node	**Description**
$out_{b\in\{1,\ldots,4\}}$	tuples v' to map each velocity in the set $v_{k\in\{l,r\}}$
$out_{b\in\{5,\ldots,8\}}$	activation of each color in the set $\{\varnothing, C, M, Y\}$
Connection	**Description**
$conn_{s\in\{1,\ldots,328\}}$	synaptic connections with weights $\omega\in[-5,5]$
Number of generations *	—
Population size	100
Elite individuals	20
Mutated individuals	80
Evaluations per individual	10
Post-evaluation per individual **	100

* The number of generations is computed according to the budget of simulations. ** The population obtained in the last generation is post-evaluated to select the best individual.

The readings of the proximity $(prox)$ and ground (gnd) sensors are passed directly to the network. Information about the number of neighboring peers (n) is provided through the function $z'(n) \in [0,1]$, with $z'(n) = 1 - \frac{2}{1+e^{(n)}}$. The vector V_n and each vector in $V_{c\in\{R,G,B,C,M,Y\}}$ are translated into scalar projections onto four unit vectors that point at 45°, 135°, 225°, and 315° with respect to the front of the robot. Then, each projection is passed to the network through an independent input node. The last input of the network comes from a bias node. Four output nodes encode tuples (v') of negative and positive components of the velocity of the wheels. Each tuple is obtained from two independent output nodes and is defined as $v' = ([-12,0],[0,12])$. The velocity of a wheel (v) is computed as the sum of the two elements in a tuple (v'). Similarly, the color displayed by the RGB LEDs of the robot is selected by comparing the value of the output nodes that correspond to colors in the set $\{\varnothing, C, M, Y\}$. The color displayed corresponds to the maximum value found across the four colors.

`EvoColor` differs from `EvoStick` in two aspects: the reference model and how the output of the neural network is mapped to the velocity of the robots. First, `EvoColor` is based on RM 3 and `EvoStick` on RM 1.1. In accordance to RM 3, `EvoColor` does not integrate the capability of the e-pucks for sensing the intensity of ambiance light—originally integrated in `EvoStick`. The second difference between `EvoColor` and `EvoStick` is how the output of the neural network is mapped to the velocity of the e-pucks. In `EvoColor`, we introduce a velocity mapping based on tuples to facilitate the evolution of standstill behaviors—as we expect robots need them to perform STOP and AGGREGATION.

In `EvoStick`, the control software maps directly a single output node of the neural network into velocity commands $(v = [-12,12])$ for each wheel $(v_{k\in\{l,r\}})$—a robot can stand still only if the velocity of the two wheels is set exactly to 0. A standstill behavior is then difficult to achieve since only one pair of values in the output nodes maps exactly to $v_l = 0$ and $v_r = 0$; Moreover, the output nodes can not maintain a steady value because they are subject to the injection of sensory noise. In `EvoColor`, the control software maps the sum of elements of a tuple (v') into velocity commands for each wheel $v_{k\in\{l,r\}}$—each tuple is defined by two output nodes and provides a negative and a

positive component to compute the velocity. We expect that this mapping facilitates the evolution of standstill behaviors: first, robots can stand still if the elements of each tuple (v') are any pair of values of equal magnitude—steady values are not required provided that the output nodes that encode the same tuple vary proportionally; second, the sum of the positive and negative components can cancel out the sensory noise injected in the output nodes that encode a tuple—given a proper tuning of the synaptic weights. If one compares `EvoColor` with `EvoStick`, the first has more freedom to tune neural networks that lead to standstill behaviors.

4.3. Protocol

We conduct experiments with twenty e-pucks on the missions described in Section 4.1. For each mission, we produce ten designs of control software with `TuttiFrutti` and ten with `EvoColor`. We assess the effectiveness of the methods by testing each design once in simulation and once with physical robots.

Statistics

We use box-plots to represent the performance of the control software we produce. For each method, we report the performance obtained in simulation (thin boxes) and with physical robots (thick ones). In all cases, we support comparative statements with an exact binomial test, at 95% confidence [66]: statements like "*A* performs *significantly* better/worse than *B*" imply that the comparison is supported by a an exact binomial test, at 95% confidence. In addition, we estimate the overall performance of `TuttiFrutti` with respect to `EvoColor`. To this purpose, we aggregate the results by comparing the performance of the two design methods across each mission. In the context of the overall performance of the design methods, any statement like "*A* performs *significantly* better/worse than *B*" also implies that the comparison is supported by an exact binomial test, at 95% confidence [66].

5. Results and Discussion

We present the qualitative and quantitative analysis of the results obtained with `TuttiFrutti` and `EvoColor`. We discuss first the behavior and performance of the swarms on a per-mission basis. Then, we elaborate on the aggregate performance across the three missions. In the end, we address the research questions presented in Section 1 and we discuss our findings. The code, control software and demonstrative videos are provided as Supplementary Material [67]. In the context of these results, references to colored robots imply that the robots display the named color—for example, "*cyan robots*" stands for robots that display the color cyan.

5.1. STOP

Figure 3 (left) shows the performance of `TuttiFrutti` and `EvoColor` in STOP. In this mission, `TuttiFrutti` performs significantly better than `EvoColor`.

From visual inspection, `TuttiFrutti` produced control software that effectively uses the capabilities of the robots for displaying and perceiving colors. The swarm first disperses and homogeneously covers the arena—aiming to rapidly detect the stop signal. If a robot detects the stop signal, it stands still and disseminates the information by emitting a signal of an arbitrary color. When other robots perceive the signal emitted by their peer, they also transition to a standstill behavior and relay the signal. The process continues until all robots in the swarm are standing still. We consider that this behavior shows the potential of `TuttiFrutti` for producing event-handling collective behaviors. The swarm collectively transitions from coverage to standstill when the stop signal appears. As we expected, `TuttiFrutti` produces control software that establishes communication protocols by correctly pairing the color of the signals that robots emit and the behavior other robots must adopt when they perceive them—similarly to the results obtained by Hasselmann et al. [14] with `Gianduja`.

`EvoColor`, unlike `TuttiFrutti`, designed collective behaviors that do not respond to the stop signal. The swarm adopts a rather simplistic behavior in which robots move until stopped by the walls. They remain then in a standstill behavior because they persistently push against the walls—no reaction can be appreciated in the swarms when the stop signal appears. This behavior was observed too in the experiments with physical robots, and in many cases, robots maintained standing-still behaviors by pushing against other robots too.

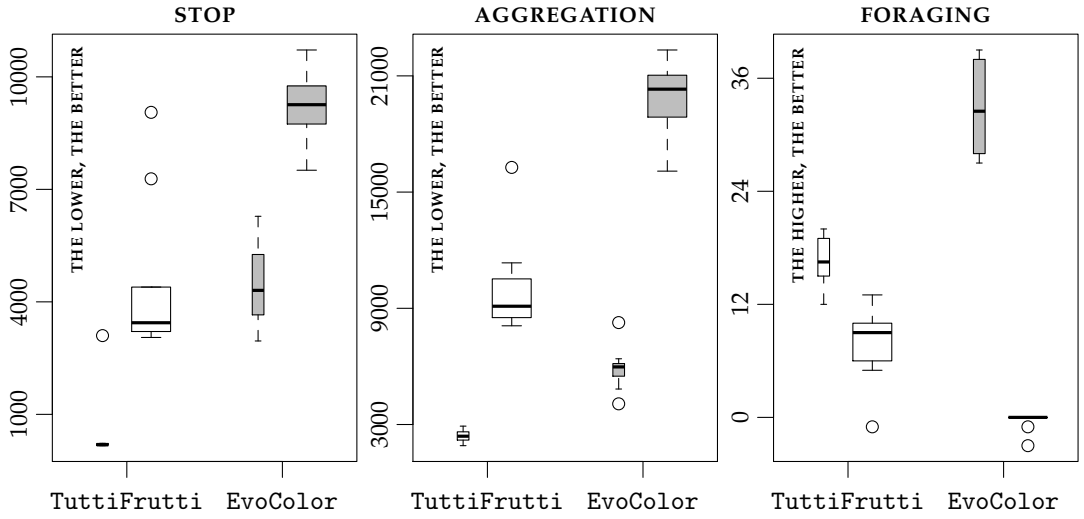

Figure 3. Performance obtained in the missions STOP (left), AGGREGATION (center), and FORAGING (right). The performance of `TuttiFrutti` is shown in white and the one of `EvoColor` in gray. Thin boxes represent results obtained in simulation and thick boxes the ones obtained with physical robots.

In the experiments with physical robots, both `TuttiFrutti` and `EvoColor` showed a significant drop in performance with respect to the simulations. However, the difference in mean performance between simulations and experiments with physical robots is larger for `EvoColor` than `TuttiFrutti`. Swarms deployed with the control software produced by `TuttiFrutti` showed the same collective behavior observed in simulation, although the rapidness in discovering the stop signal and disseminating the information decreased. In the case of `EvoColor`, robots often do not reach the walls and they push against each other to remain still in place.

Figure 4 shows an example of the control software produced by `TuttiFrutti` for STOP. Robots start in REPULSION with no color displayed ($\gamma = \varnothing$). They transition to STOP and turn yellow ($\gamma = Y$) when COLOR-DETECTION is triggered either by a green wall ($\delta = G$) or by yellow robots ($\delta = Y$). In this sense, robots change their behavior when they either perceive the stop signal or the yellow signals that other robots emit.

5.2. AGGREGATION

Figure 3 (center) shows the performance of `TuttiFrutti` and `EvoColor` in AGGREGATION. In this mission, `TuttiFrutti` performs significantly better than `EvoColor`.

Also in this case, from visual inspection, `TuttiFrutti` produced control software that effectively uses the capabilities that the robots have of displaying and perceiving colors. As we expected, `TuttiFrutti` designs collective behaviors in which robots reach and remain in the blue zone by moving towards blue walls. This behavior is often complemented with navigation or communication strategies that boost the efficiency of the swarm. For example, some instances of control software include a repulsion behavior that drives robots away from the green walls—robots reach the blue zone faster by avoiding unnecessary exploration in the green zone. In other instances, robots that step in the blue zone, or perceive the blue walls, emit a signal of an arbitrary color—other robots then follow this

signal to reach the blue zone. In this sense, robots communicate and collectively navigate to aggregate faster. Finally, some instances combine the two strategies.

`EvoColor` designed collective behaviors in which robots use the colors displayed in the arena. Robots explore the arena until they step in one of the black regions—either at the blue or green zone. If robots step in the green zone, they move away from the green walls and reach the blue zone. If robots step in the blue zone, they attempt to stand still. In this sense, robots react and avoid the green walls as a strategy to aggregate in the blue zone.

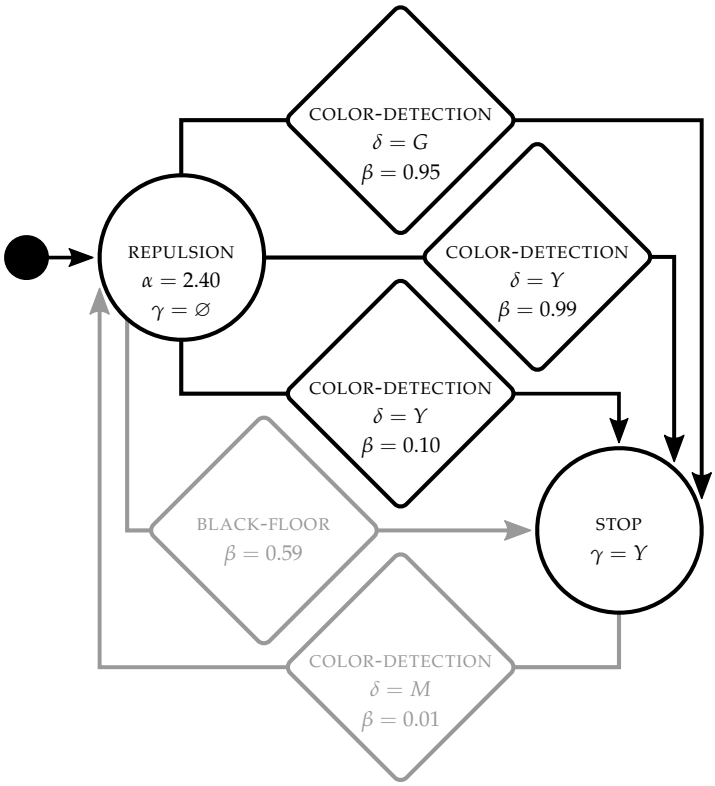

Figure 4. Instance of control software produced by `TuttiFrutti` for STOP. The probabilistic finite state machine shows the effective modules in black and non-reachable modules in light gray. Circular modules represent the low-level behaviors and rhomboid modules represent transition conditions.

The control software produced by `TuttiFrutti` and `EvoColor` showed a significant drop in performance when ported to the physical robots. As observed in STOP, the difference in mean performance between simulations and experiments with physical robots is larger for `EvoColor` than `TuttiFrutti`. Robot swarms that use the control software produced by `TuttiFrutti` display the same collective behaviors observed in simulation. The decrease in performance occurs because few robots that leave the blue zone do not return as fast as observed in the simulations. The control software produced by `EvoColor` does not port well to the physical robots—that is, robots appear to be unable to reproduce the behaviors observed in the simulation. Robots ramble in the arena and seem to react to the presence of their peers, however, no specific meaningful behavior could be identified by visual inspection.

Figure 5 shows an example of the control software produced by `TuttiFrutti` for AGGREGATION. Robots start in COLOR-FOLLOWING displaying yellow ($\delta = Y$) and move towards cyan robots ($\gamma = C$). When they perceive the blue walls ($\delta = B$), COLOR-DETECTION triggers and the robots transition to a second module COLOR-FOLLOWING in which they move towards the blue walls ($\delta = B$) while emitting a cyan signal ($\gamma = C$). By cycling in these behaviors, robots can navigate to the blue zone either by moving towards the blue walls or by following the cyan signals that other robots emit. The transition

conditions FIXED-PROBABILITY, GRAY-FLOOR and NEIGHBOR-COUNT trigger the COLOR-FOLLOWING behavior that allows the robot to return to the aggregation area.

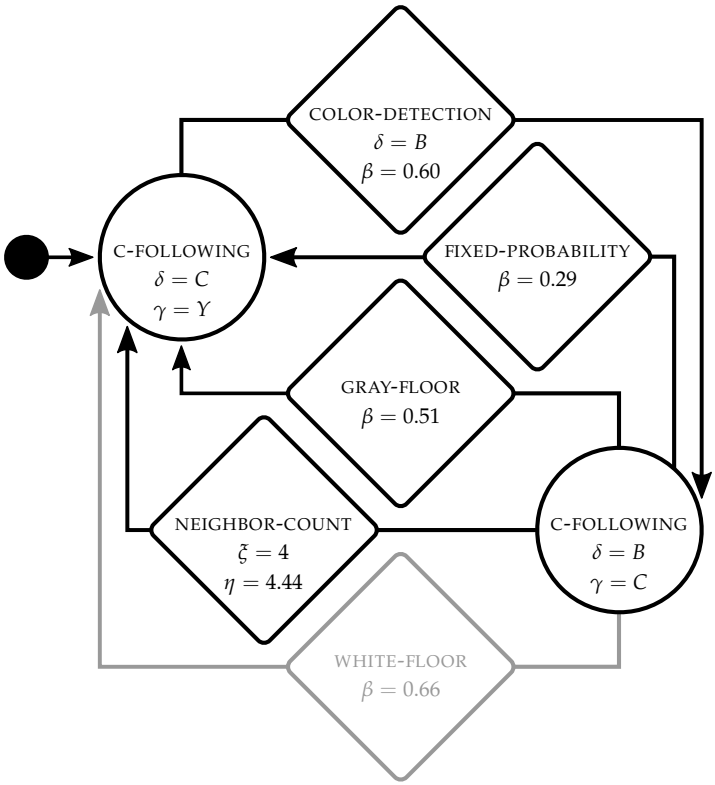

Figure 5. Instance of control software produced by `TuttiFrutti` for AGGREGATION. The probabilistic finite state machine shows the effective modules in black and non-reachable modules in light gray. Circular modules represent the low-level behaviors and rhomboid modules represent the transition conditions. Modules labeled as C-FOLLOWING stand for the low-level behavior COLOR-FOLLOWING.

5.3. FORAGING

Figure 3 (right) shows the performance of `TuttiFrutti` and `EvoColor` in FORAGING. In this mission, `EvoColor` performs significantly better than `TuttiFrutti` in simulation. However, `TuttiFrutti` performs significantly better than `EvoColor` in the experiments with physical robots.

As in the other missions, from visual inspection, `TuttiFrutti` produced control software that effectively uses the capabilities that the robots have of displaying and perceiving colors. Robots explore the arena and forage only from the profitable source. However, contrary to what we expected, `TuttiFrutti` designed collective behaviors that do not use all the colors displayed in the arena. In fact, robots mostly forage by randomly exploring the arena while moving away from the green wall—in other words, they only avoid to step in the green source. Although the swarm can perform the mission with this behavior, we expected that robots could navigate faster by moving towards the blue and red walls. Still, `TuttiFrutti` produced only few instances of control software in which robots react to more than one color—see Figure 6. We conjecture that `TuttiFrutti` exploits the convex shape of the arena to produce solutions that are effective at the minimal complexity—that is, the performance of a swarm in this mission might not improve even if robots react to all three colors.

`EvoColor` designed collective behaviors in which the swarm does not react to the colors displayed in the arena. Robots forage from the blue source by following the walls of the arena in a clockwise direction. This behavior efficiently drives the robots around the arena and across the blue source. When the robots reach the intersection that divides the blue and green source, they continue moving straight and effectively reach the nest. By cycling in this behavior, the swarm maintains an efficient stream of foraging robots.

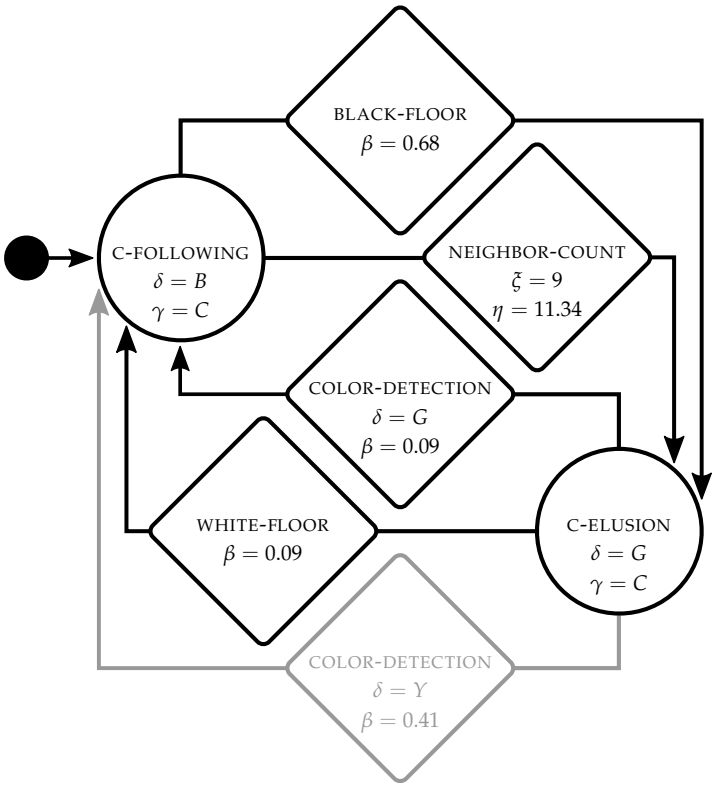

Figure 6. Instance of control software produced by `TuttiFrutti` for FORAGING. The probabilistic finite state machine shows the effective modules in black and non-reachable modules in light gray. Circular modules represent the low-level behaviors and rhomboid modules represent the transition conditions. Modules labeled as C-FOLLOWING and C-ELUSION stand for the low-level behaviors COLOR-FOLLOWING and COLOR-ELUSION, respectively.

`TuttiFrutti` and `EvoColor` showed a significant drop in performance in the experiments with physical robots, in comparison to the performance obtained in the simulations. Likewise the other two missions, the difference in mean performance between simulations and experiments with physical robots is larger for `EvoColor` than `TuttiFrutti`. In the case of `TuttiFrutti`, we did not observe any difference in the behavior of the swarms with respect to the simulations. Conversely, the collective behaviors designed by `EvoColor` are affected to the point that the swarm is unable to complete the mission. In the control software produced by `EvoColor`, the ability of the robots to follow the walls strongly depends on the fine-tuning of the synaptic weights in the neural network—more precisely, it requires a precise mapping between the proximity sensors and wheels of the robots. In the physical robots, the noise of the proximity sensors and wheels differs from the original design model, and a fine-tuned neural network is less effective. Indeed, the swarm is not any more able to maintain the stream of foraging robots, and on the contrary, robots stick to each other and to the walls.

We also observe a *rank inversion* of the performance of the two methods in this mission. As defined by Ligot and Birattari [53], a rank inversion is a phenomenon that manifests when an instance of control software outperforms another in simulation, but it is outperformed by the latter when it is evaluated on physical robots. In our experiments, `TuttiFrutti` is outperformed by `EvoColor` in simulation, but it outperforms `EvoColor` when it is ported to the physical robots. These results are consistent with the ones reported by Francesca et al. [8], and further discussed by Birattari et al. [68] and Ligot and Birattari [53], for comparisons between the modular and the neuro-evolutionary approach to the automatic design of robot swarms.

Figure 6 shows an example of the control software produced by `TuttiFrutti` for FORAGING. Robots start in COLOR-FOLLOWING displaying cyan ($\gamma = C$) and moving towards the blue wall ($\delta = B$). If a robot steps in one of the two sources, BLACK-FLOOR triggers and the robot

transitions to COLOR-ELUSION—it then becomes cyan ($\gamma = C$) and moves away from the green wall ($\delta = G$). When the robot steps in the nest, WHITE-FLOOR triggers and the robot transitions back to COLOR-FOLLOWING. By cycling this behavior, robots move back and forth between the blue source and the nest. When robots are in COLOR-ELUSION, COLOR-DETECTION can trigger with a low probability ($\beta = 0.09$) if robots perceive the green wall ($\delta = G$). This transition mitigates the penalty caused by robots that step in the green source. If a robot steps in the green source, it transitions back to COLOR-FOLLOWING and moves towards the blue wall. Finally, the transition condition NEIGHBOR-COUNT can trigger when the robot perceives more than four neighboring robots. Yet, we do not find a clear effect of this transition in the overall behavior of the robots.

5.4. Aggregate Results

`TuttiFrutti` and `EvoColor` obtain similar results when the control software is evaluated with simulations. On the other hand, `TuttiFrutti` is significantly better than `EvoColor` when the control software is ported to the physical robots. It has already been pointed out that when control software developed in simulation is ported to a real-world platform, due to the reality gap one might observe both a drop in performance [53] and a substantial modification of the collective behavior [69]. The entity of these effects might depend on the design method, and some design methods might be more robust than others [53]. Our results indicate that `EvoColor` is more affected by the reality gap than `TuttiFrutti` across the three missions considered. This is apparent both in the entity of the performance drop we measured and in the fact that the collective behaviors of the control software generated by `EvoColor` are often dramatically differently in simulation and in the real world, while the ones of the control software generated by `TuttiFrutti` are essentially unchanged.

By introducing `TuttiFrutti`, we also investigated the impact of an extended design space in the optimization process of AutoMoDe. The size of the design space in `Vanilla` and `Chocolate` is $O(|B|^4\,|C|^{16})$, as estimated by Kuckling et al. [70]. B and C represent, respectively, the number of modules in low-level behaviors and transition conditions. Using the same method as Kuckling et al., we estimate the design space in `TuttiFrutti` to be $O(|4B|^4\,|C|^{16})$—that is, 256 times larger than the one searched by `Chocolate`. Notwithstanding the larger design space, we do not find evidence that `TuttiFrutti` is affected by the increased number of parameters to tune. Indeed, `TuttiFrutti` produced effective control software for all missions considered.

5.5. Discussion

In the following, we first address the research questions defined in Section 1 and then we discuss our findings.

`TuttiFrutti` selects, tunes and assembles control software that operates with information that is available in the environment in the form of colors. In the three missions, the robot swarm reacts to these colors and act according to the information they provide in each case—both for handling events and navigating. Additionally, we observed that `TuttiFrutti` can design collective behaviors that exhibit color-based communication between robots. For example, `TuttiFrutti` designed collective behaviors with color-based communication in STOP and AGGREGATION—missions in which communication can influence the performance of the robot swarm. These collective behaviors are feasible thanks to the extended capabilities of the e-puck, capabilities that translated into a larger space of possible control software than the one considered by `Vanilla` and `Chocolate`—early versions of AutoMoDe. As the design space of `TuttiFrutti` is larger than the one of `Vanilla` and `Chocolate`, one could have expected that the automatic design process would have difficulties in producing meaningful control software. Still, we did not find evidence that `TuttiFrutti` suffers from an increased difficulty to design collective behaviors for robot swarms. The reference model RM 3 and the set of modules introduced with `TuttiFrutti` allowed it to conceive STOP and AGGREGATION—variants of missions already studied with AutoMoDe, and FORAGING—a new mission framed within the *best-of-n* problem.

By introducing TuttiFrutti, we enlarged the variety of collective behaviors that can be produced with the AutoMoDe family.

We argue that the experiments we conducted with TuttiFrutti show evidence that automatic modular design methods can establish a mission-specific relationship between the colors that the robots perceive and the behavior that they must adopt. In Section 2, we described experiments in which this relationship enabled the design of complex collective behaviors [25,31,35]. We find that these collective behaviors have similarities with those designed by TuttiFrutti—for example, robots react to colored objects in the environment and use colors signals to communicate with their peers. We conjecture that TuttiFrutti, or design methods that might share its characteristics, can produce a wider range and more complex collective behaviors than those described in this paper. In this sense, we believe that research with robot swarms that can perceive and display colors has the potential to close the gap between the complexity of the missions performed with manual design methods, and those performed with automatic design.

6. Conclusions

In this paper, we introduced AutoMoDe-TuttiFrutti—an automatic method to design collective behaviors for robots that can perceive and communicate color-based information. We designed control software for swarms of e-pucks that comply with RM 3—e-pucks can use their LEDs to display colors and their omnidirectional vision turret to perceive them. The capability of the robots to act upon different colors translated into an increased variety of collective behaviors compared to previous instances of AutoMoDe. We assessed TuttiFrutti on a class of missions in which the performance of the swarm depends on its ability to use color-based information for handling events, communicating, and navigating.

We conducted experiments in simulation and with physical robot swarms performing three missions: STOP, AGGREGATION and FORAGING. In all cases, TuttiFrutti designed collective behaviors that effectively use color-based information. In STOP, the swarm collectively changes its behavior when a specific color signal appears. In STOP and AGGREGATION, the swarm exhibits communication behaviors in which robots pair the color signals they emit and the colors to which they react. In AGGREGATION and FORAGING, robots use the colors they perceive as a reference to navigate the environment. In FORAGING, swarms differentiate two sources of items and forage from the profitable one. Alongside the results obtained with TuttiFrutti, we assessed a method based on neuro-evolution: EvoColor. In STOP and FORAGING, EvoColor designed collective behaviors that do not use color-based information. In AGGREGATION, EvoColor designed collective behaviors in which robots use the colors they perceive to navigate the environment—likewise TuttiFrutti. The aggregated results showed that TuttiFrutti performs better than EvoColor in the class of missions we considered. Results with physical robots suggest that TuttiFrutti crosses the reality gap better than EvoColor—result partially sustained by the visual inspection of the behavior of the robots.

Automatic design methods can effectively produce control software for swarms of robots that can display and perceive colors. We demonstrated that TuttiFrutti establishes an appropriate relationship between the colors that the robots perceive and the behavior they must adopt. In our experiments, this relationship was established on a per-mission basis and responded to the specifications of each mission. Yet, the set of missions on which we assess TuttiFrutti is far from being exhaustive, and more research work is needed to define the limitations of the design method. Future work will be devoted to assess TuttiFrutti in a larger and more complex class of missions. It is our contention that TuttiFrutti can design collective behaviors to address missions that involve a larger number of features in the environment and time-varying conditions. As observed in STOP, robots can effectively transition between two collective behaviors. We foresee that this ability enables the design of swarms that can perform missions with two or more sequential tasks. To the best of our knowledge, the design of collective behaviors to address this class of missions has not been studied in the context of automatic off-line design of robot swarms.

Supplementary Materials: The code, control software and demonstrative videos are available online at http://iridia.ulb.ac.be/supp/IridiaSupp2019-008.

Author Contributions: Implementations and experiments were done by D.G.R. The paper was drafted by D.G.R. and refined by M.B. The research was directed by M.B. All authors have read and agreed to the published version of the manuscript.

Funding: The project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 681872). M.B. acknowledges support from the Belgian *Fonds de la Recherche Scientifique*—FNRS. D.G.R. acknowledges support from the Colombian Administrative Department of Science, Technology and Innovation—COLCIENCIAS.

Acknowledgments: The authors thank Federico Pagnozzi and Jonas Kuckling for reading a preliminary version of this paper.

Conflicts of 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. Beni, G. From swarm intelligence to swarm robotics. In *Swarm Robotics, Proceedings of the SAB 2004 International Workshop, Santa Monica, CA, USA, 17 July 2004*; Şahin, E., Spears, W.M., Eds.; LNCS; Springer: Berlin, Germany, 2004; Volume 3342, pp. 1–9. [CrossRef]
2. Şahin, E. Swarm robotics: From sources of inspiration to domains of application. In *Swarm Robotics, Proceedings of the SAB 2004 International Workshop, Santa Monica, CA, USA, 17 July 2004*; Şahin, E., Spears, W.M., Eds.; LNCS; Springer: Berlin, Germany, 2004; Volume 3342, pp. 10–20. [CrossRef]
3. Dorigo, M.; Birattari, M.; Brambilla, M. Swarm robotics. *Scholarpedia* **2014**, *9*, 1463. [CrossRef]
4. Brambilla, M.; Ferrante, E.; Birattari, M.; Dorigo, M. Swarm robotics: A review from the swarm engineering perspective. *Swarm Intell.* **2013**, *7*, 1–41. [CrossRef]
5. Francesca, G.; Birattari, M. Automatic design of robot swarms: Achievements and challenges. *Front. Robot. AI* **2016**, *3*, 1–9. [CrossRef]
6. Nolfi, S.; Floreano, D. *Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines*, 1st ed.; A Bradford Book; MIT Press: Cambridge, MA, USA, 2000.
7. Trianni, V. *Evolutionary Swarm Robotics*; Springer: Berlin, Germany, 2008. [CrossRef]
8. Francesca, G.; Brambilla, M.; Brutschy, A.; Trianni, V.; Birattari, M. AutoMoDe: A novel approach to the automatic design of control software for robot swarms. *Swarm Intell.* **2014**, *8*, 89–112. [CrossRef]
9. Francesca, G.; Brambilla, M.; Brutschy, A.; Garattoni, L.; Miletitch, R.; Podevijn, G.; Reina, A.; Soleymani, T.; Salvaro, M.; Pinciroli, C.; et al. AutoMoDe-Chocolate: Automatic design of control software for robot swarms. *Swarm Intell.* **2015**, *9*, 125–152. [CrossRef]
10. Kuckling, J.; Ligot, A.; Bozhinoski, D.; Birattari, M. Behavior trees as a control architecture in the automatic modular design of robot swarms. In *Swarm Intelligence, Proceedings of the 11th International Conference, ANTS 2018, Rome, Italy, 29–31 October 2018*; Dorigo, M., Birattari, M., Blum, C., Christensen, A.L., Reina, A., Trianni, V., Eds.; LNCS; Springer: Cham, Switzerland, 2018; Volume 11172, pp. 30–43. [CrossRef]
11. Salman, M.; Ligot, A.; Birattari, M. Concurrent design of control software and configuration of hardware for robot swarms under economic constraints. *PeerJ Comput. Sci.* **2019**, *5*, e221. [CrossRef]
12. Spaey, G.; Kegeleirs, M.; Garzón Ramos, D.; Birattari, M. Comparison of different exploration schemes in the automatic modular design of robot swarms. In Proceedings of the Reference AI & ML Conference for Belgium, Netherlands & Luxemburg, BNAIC/BENELEARN, Brussels, Belgium, 6–8 November 2019; CEUR Workshop Proceedings; Beuls, K., Bogaerts, B., Bontempi, G., Geurts, P., Harley, N., Lebichot, B., Lenaerts, T., Gilles, L., Van Eecke, P., Eds.; CEUR-WS.org: Aachen, Germany, 2019; Volume 2491.
13. Kuckling, J.; Ubeda Arriaza, K.; Birattari, M. Simulated annealing as an optimization algorithm in the automatic modular design of robot swarms. In Proceedings of the Reference AI & ML Conference for Belgium, Netherlands & Luxemburg, BNAIC/BENELEARN, Brussels, Belgium, 6–8 November 2019; CEUR Workshop Proceedings; Beuls, K., Bogaerts, B., Bontempi, G., Geurts, P., Harley, N., Lebichot, B., Lenaerts, T., Gilles, L., Van Eecke, P., Eds.; CEUR-WS.org: Aachen, Germany, 2019; Volume 2491.

14. Hasselmann, K.; Robert, F.; Birattari, M. Automatic design of communication-based behaviors for robot swarms. In *Swarm Intelligence, Proceedings of the 11th International Conference, ANTS 2018, Rome, Italy, 29–31 October 2018*; Dorigo, M., Birattari, M., Blum, C., Christensen, A.L., Reina, A., Trianni, V., Eds.; LNCS; Springer: Cham, Switzerland, 2018; Volume 11172, *LNCS*, pp. 16–29. [CrossRef]
15. Francesca, G.; Brambilla, M.; Brutschy, A.; Garattoni, L.; Miletitch, R.; Podevijn, G.; Reina, A.; Soleymani, T.; Salvaro, M.; Pinciroli, C.; et al. An experiment in automatic design of robot swarms: AutoMoDe-Vanilla, EvoStick, and human experts. In *Swarm Intelligence, Proceedings of the 9th International Conference, ANTS 2014, Brussels, Belgium, 10–12 September 2014*; Dorigo, M., Birattari, M., Garnier, S., Hamann, H., Montes de Oca, M., Solnon, C., Stützle, T., Eds.; LNCS; Springer International Publishing: Berlin, Germany, 2014; Volume 8667, pp. 25–37. [CrossRef]
16. École Polytechnique Fédérale de Lausanne. Omnidirectional Vision Turret for the e-Puck. 2010. Available online: http://www.e-puck.org/index.php?option=com_content&view=article&id=26&Itemid=21 (accessed on 3 July 2020).
17. Birattari, M.; Ligot, A.; Bozhinoski, D.; Brambilla, M.; Francesca, G.; Garattoni, L.; Garzón Ramos, D.; Hasselmann, K.; Kegeleirs, M.; Kuckling, J.; et al. Automatic off-line design of robot swarms: A manifesto. *Front. Robot. AI* **2019**, *6*, 59. [CrossRef]
18. Waibel, M.; Keller, L.; Floreano, D. Genetic team composition and level of selection in the evolution of multi-agent systems. *IEEE Trans. Evol. Comput.* **2009**, *13*, 648–660. [CrossRef]
19. Gauci, M.; Chen, J.; Li, W.; Dodd, T.J.; Groß, R. Self-organized aggregation without computation. *Int. J. Robot. Res.* **2014**, *33*, 1145–1161. [CrossRef]
20. Chen, J.; Gauci, M.; Li, W.; Kolling, A.; Groß, R. Occlusion-based cooperative transport with a swarm of miniature mobile robots. *IEEE Trans. Robot.* **2015**, *31*, 307–321. [CrossRef]
21. Lopes, Y.K.; Trenkwalder, S.M.; Leal, A.B.; Dodd, T.J.; Groß, R. Supervisory control theory applied to swarm robotics. *Swarm Intell.* **2016**, *10*, 65–97. [CrossRef]
22. Jones, S.; Winfield, A.; Hauert, S.; Studley, M. Onboard evolution of understandable swarm behaviors. *Adv. Intell. Syst.* **2019**, *1*, 1900031. [CrossRef]
23. O'Grady, R.; Christensen, A.L.; Dorigo, M. SWARMORPH: Multirobot morphogenesis using directional self-assembly. *IEEE Trans. Robot.* **2009**, *25*, 738–743. [CrossRef]
24. O'Grady, R.; Groß, R.; Christensen, A.L.; Dorigo, M. Self-assembly strategies in a group of autonomous mobile robots. *Auton. Robot.* **2010**, *28*, 439–455. [CrossRef]
25. Mathews, N.; Christensen, A.L.; O'Grady, R.; Mondada, F.; Dorigo, M. Mergeable nervous systems for robots. *Nat. Commun.* **2017**, *8*, 439. [CrossRef] [PubMed]
26. Mathews, N.; Christensen, A.L.; Stranieri, A.; Scheidler, A.; Dorigo, M. Supervised morphogenesis: Exploiting morphological flexibility of self-assembling multirobot systems through cooperation with aerial robots. *Robot. Auton. Syst.* **2019**, *112*, 154–167. [CrossRef]
27. Christensen, A.L.; O'Grady, R.; Dorigo, M. From fireflies to fault-tolerant swarms of robots. *IEEE Trans. Evol. Comput.* **2009**, *13*, 754–766. [CrossRef]
28. Nouyan, S.; Groß, R.; Bonani, M.; Mondada, F.; Dorigo, M. Teamwork in self-organized robot colonies. *IEEE Trans. Evol. Comput.* **2009**, *13*, 695–711. [CrossRef]
29. Ducatelle, F.; Di Caro, G.A.; Pinciroli, C.; Gambardella, L.M. Self-organized cooperation between robotic swarms. *Swarm Intell.* **2011**, *5*, 73–96. [CrossRef]
30. Dorigo, M.; Floreano, D.; Gambardella, L.M.; Mondada, F.; Nolfi, S.; Baaboura, T.; Birattari, M.; Bonani, M.; Brambilla, M.; Brutschy, A.; et al. Swarmanoid: A novel concept for the study of heterogeneous robotic swarms. *IEEE Robot. Autom. Mag.* **2013**, *20*, 60–71. [CrossRef]
31. Garattoni, L.; Birattari, M. Autonomous task sequencing in a robot swarm. *Sci. Robot.* **2018**, *3*, eaat0430. [CrossRef]
32. Ferrante, E.; Turgut, A.E.; Mathews, N.; Birattari, M.; Dorigo, M. Flocking in stationary and non-stationary environments: A novel communication strategy for heading alignment. In *Parallel Problem Solving from Nature, PPSN XI*; Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2010; Volume 6239, pp. 331–340. [CrossRef]

33. Giusti, A.; Nagi, J.; Gambardella, L.M.; Di Caro, G.A. Distributed consensus for interaction between humans and mobile robot swarms (demonstration). In *AAMAS '12: Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems—Volume 3*; International Foundation for Autonomous Agents and Multiagent Systems: Richland, SC, USA, 2012; pp. 1503–1504.
34. Podevijn, G.; O'Grady, R.; Dorigo, M. Self-organised feedback in human swarm interaction. In Proceedings of the Workshop on Robot Feedback in Human-Robot Interaction: How to Make a Robot Readable for a Human Interaction Partner, Ro-Man 2012, Paris, France, 9 September 2012.
35. Nouyan, S.; Campo, A.; Dorigo, M. Path formation in a robot swarm: Self-organized strategies to find your way home. *Swarm Intell.* **2008**, *2*, 1–23. [CrossRef]
36. Pinciroli, C.; O'Grady, R.; Christensen, A.L.; Dorigo, M. Self-organised recruitment in a heteregeneous swarm. In Proceedings of the 2009 International Conference on Advanced Robotics, (ICAR), Munich, Germany, 22–26 June 2009; IEEE: Piscataway, NJ, USA, 2009; pp. 1–8.
37. Pini, G.; Brutschy, A.; Birattari, M.; Dorigo, M. Task partitioning in swarms of robots: Reducing performance losses due to interference at shared resources. In *Informatics in Control Automation and Robotics*; Andrade Cetto, J., Filipe, J., Ferrier, J.L., Eds.; Lecture Notes in Electrical Engineering; Springer: Berlin/Heidelberg, Germany, 2009; Volume 85, pp. 217–228. [CrossRef]
38. Pini, G.; Brutschy, A.; Frison, M.; Roli, A.; Dorigo, M.; Birattari, M. Task partitioning in swarms of robots: An adaptive method for strategy selection. *Swarm Intell.* **2011**, *5*, 283–304. [CrossRef]
39. Pini, G.; Brutschy, A.; Scheidler, A.; Dorigo, M.; Birattari, M. Task partitioning in a robot swarm: Object retrieval as a sequence of subtasks with direct object transfer. *Artif. Life* **2014**, *20*, 291–317. [CrossRef]
40. Brutschy, A.; Garattoni, L.; Brambilla, M.; Francesca, G.; Pini, G.; Dorigo, M.; Birattari, M. The TAM: Abstracting complex tasks in swarm robotics research. *Swarm Intell.* **2015**, *9*, 1–22. [CrossRef]
41. Allwright, M.; Bhalla, N.; El-faham, H.; Antoun, A.; Pinciroli, C.; Dorigo, M. SRoCS: Leveraging stigmergy on a multi-robot construction platform for unknown environments. In *Swarm Intelligence, Proceedings of the 9th International Conference, ANTS 2014, Brussels, Belgium, 10–12 September 2014*; Dorigo, M., Birattari, M., Garnier, S., Hamann, H., Montes de Oca, M., Solnon, C., Stützle, T., Eds.; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2014; Volume 8667, pp. 158–169. [CrossRef]
42. Brambilla, M.; Brutschy, A.; Dorigo, M.; Birattari, M. Property-driven design for swarm robotics: A design method based on prescriptive modeling and model checking. *ACM Trans. Auton. Adapt. Syst.* **2014**, *9*, 17:1–17:28. [CrossRef]
43. Floreano, D.; Mitri, S.; Magnenat, S.; Keller, L. Evolutionary conditions for the emergence of communication in robots. *Curr. Biol.* **2007**, *17*, 514–519. [CrossRef]
44. Ampatzis, C.; Tuci, E.; Trianni, V.; Christensen, A.L.; Dorigo, M. Evolving self-assembly in autonomous homogeneous robots: Experiments with two physical robots. *Artif. Life* **2009**, *15*, 465–484. [CrossRef]
45. Sperati, V.; Trianni, V.; Nolfi, S. Evolving coordinated group behaviours through maximisation of mean mutual information. *Swarm Intell.* **2008**, *2*, 73–95. [CrossRef]
46. Sperati, V.; Trianni, V.; Nolfi, S. Self-organised path formation in a swarm of robots. *Swarm Intell.* **2011**, *5*, 97–119. [CrossRef]
47. Trianni, V.; López-Ibáñez, M. Advantages of task-specific multi-objective optimisation in evolutionary robotics. *PLoS ONE* **2015**, *10*, e0136406. [CrossRef]
48. Nedjah, N.; Silva Junior, L. Review of methodologies and tasks in swarm robotics towards standardization. *Swarm Evol. Comput.* **2019**, *50*, 100565. [CrossRef]
49. Mayet, R.; Roberz, J.; Schmickl, T.; Crailsheim, K. Antbots: A feasible visual emulation of pheromone trails for swarm robots. In *Swarm Intelligence, Proceedings of the 7th International Conference, ANTS 2010, Brussels, Belgium, 8–10 September 2010*; Dorigo, M., Birattari, M., Di Caro, G.A., Doursat, R., Engelbrecht, A.P., Floreano, D., Gambardella, L.M., Groß, R., Şahin, E., Sayama, H., et al., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2010; Volume 6234, pp. 89–94. [CrossRef]
50. Brutschy, A.; Pini, G.; Decugnière, A. *Grippable Objects for the Foot-Bot*; Technical Report TR/IRIDIA/2012-001; IRIDIA, Université Libre de Bruxelles: Brussels, Belgium, 2012.
51. Soleymani, T.; Trianni, V.; Bonani, M.; Mondada, F.; Dorigo, M. Bio-inspired construction with mobile robots and compliant pockets. *Robot. Auton. Syst.* **2015**, *74*, 340–350, doi:10.1016/j.robot.2015.07.018. [CrossRef]
52. Kolling, A.; Walker, P.; Chakraborty, N.; Sycara, K.; Lewis, M. Human interaction with robot swarms: A survey. *IEEE Trans. Hum.-Mach. Syst.* **2016**, *46*, 9–26. [CrossRef]

53. Ligot, A.; Birattari, M. Simulation-only experiments to mimic the effects of the reality gap in the automatic design of robot swarms. *Swarm Intell.* **2019**, 1–24. [CrossRef]
54. Mondada, F.; Bonani, M.; Raemy, X.; Pugh, J.; Cianci, C.; Klaptocz, A.; Magnenat, S.; Zufferey, J.C.; Floreano, D.; Martinoli, A. The e-puck, a robot designed for education in engineering. In Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions, Castelo Branco, Portugal, 7 May 2009; Gonçalves, P., Torres, P., Alves, C., Eds.; Instituto Politécnico de Castelo Branco: Castelo Branco, Portugal, 2009; pp. 59–65.
55. Garattoni, L.; Francesca, G.; Brutschy, A.; Pinciroli, C.; Birattari, M. *Software Infrastructure for E-Puck (and TAM)*; Technical Report TR/IRIDIA/2015-004; IRIDIA, Université libre de Bruxelles: Brussels, Belgium, 2015.
56. Gutiérrez, Á.; Campo, A.; Dorigo, M.; Donate, J.; Monasterio-Huelin, F.; Magdalena, L. Open e-puck range & bearing miniaturized board for local communication in swarm robotics. In Proceedings of the IEEE International Conference on Robotics and Automation, ICRA, Kobe, Japan, 12–17 May 2009; Kosuge, K., Ed.; IEEE: Piscataway, NJ, USA, 2009; pp. 3111–3116. [CrossRef]
57. Spears, W.M.; Gordon, D.F. Using artificial physics to control agents. In Proceedings of the 1999 International Conference on Information Intelligence and Systems, Bethesda, MD, USA, 31 October–3 November 1999; IEEE Computer Society Press: Los Alamitos, CA, USA, 1999; pp. 281–288. [CrossRef]
58. Hasselmann, K.; Ligot, A.; Francesca, G.; Birattari, M. *Reference Models for AutoMoDe*; Technical Report TR/IRIDIA/2018-002; IRIDIA, Université libre de Bruxelles: Brussels, Belgium, 2018.
59. Borenstein, J.; Koren, Y. Real-time obstacle avoidance for fast mobile robots. *IEEE Trans. Syst. Man Cybern.* **1989**, *19*, 1179–1187. [CrossRef]
60. López-Ibáñez, M.; Dubois-Lacoste, J.; Pérez Cáceres, L.; Birattari, M.; Stützle, T. The irace package: Iterated racing for automatic algorithm configuration. *Oper. Res. Perspect.* **2016**, *3*, 43–58. [CrossRef]
61. Birattari, M.; Stützle, T.; Paquete, L.; Varrentrapp, K. A racing algorithm for configuring metaheuristics. In Proceedings of the GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference, New York, NY, USA, 9–13 July 2002; Langdon, W.B., Cantú-Paz, E., Mathias, K., Roy, R., Davis, D., Poli, R., Balakrishnan, K., Honavar, V., Rudolph, G., Wegener, J., et al., Eds.; Morgan Kaufmann Publishers: San Francisco, CA, USA, 2002; pp. 11–18.
62. Pinciroli, C.; Trianni, V.; O'Grady, R.; Pini, G.; Brutschy, A.; Brambilla, M.; Mathews, N.; Ferrante, E.; Di Caro, G.A.; Ducatelle, F.; et al. ARGoS: A modular, parallel, multi-engine simulator for multi-robot systems. *Swarm Intell.* **2012**, *6*, 271–295. [CrossRef]
63. Valentini, G.; Ferrante, E.; Dorigo, M. The best-of-n problem in robot swarms: Formalization, state of the art, and novel perspectives. *Front. Robot. AI* **2017**, *4*, 9. [CrossRef]
64. Valentini, G.; Hamann, H.; Dorigo, M. Efficient decision-making in a self-organizing robot swarm: On the speed versus accuracy trade-off. In Proceedings of the AAMAS '15: Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, Istanbul, Turkey, 4–8 May 2015; International Foundation for Autonomous Agents and Multiagent Systems: Richland, SC, USA, 2015; pp. 1305–1314.
65. Ligot, A.; Hasselmann, K.; Delhaisse, B.; Garattoni, L.; Francesca, G.; Birattari, M. *AutoMoDe, NEAT, and EvoStick: Implementations for the e-Puck Robot in ARGoS3*; Technical Report TR/IRIDIA/2017-002'; IRIDIA, Université libre de Bruxelles: Belgium, Brussel, 2017.
66. Conover, W.J. *Practical Nonparametric Statistics*, 3rd ed.; Wiley Series in Probability and Statistics, Applied Probability and Statistics Section; John Wiley & Sons: New York, NY, USA, 1999.
67. Garzón Ramos, D.; Birattari, M. Automatic Design of Collective Behaviors for Robots That Can Display and Perceive Colors: Supplementary Material. 2019. Available online: http://iridia.ulb.ac.be/supp/IridiaSupp2019-008 (accessed on 3 July 2020).
68. Birattari, M.; Delhaisse, B.; Francesca, G.; Kerdoncuff, Y. Observing the effects of overdesign in the automatic design of control software for robot swarms. In *Swarm Intelligence, Proceedings of the 10th International Conference, ANTS 2016, Brussels, Belgium, 7–9 September 2016*; Dorigo, M., Birattari, M., Li, X., López-Ibáñez, M., Ohkura, K., Pinciroli, C., Stützle, T., Eds.; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2016; Volume 9882, pp. 45–57. [CrossRef]

69. Floreano, D.; Husbands, P.; Nolfi, S. Evolutionary robotics. In *Springer Handbook of Robotics*, 1st ed.; Siciliano, B., Khatib, O., Eds.; Springer Handbooks; Springer: Berlin/Heidelberg, Germany, 2008; pp. 1423–1451. [CrossRef]
70. Kuckling, J.; Ligot, A.; Bozhinoski, D.; Birattari, M. *Search Space for AutoMoDe-Chocolate and AutoMoDe-Maple*; Technical Report TR/IRIDIA/2018-012; IRIDIA, Université Libre de Bruxelles: Brussels, Belgium, 2018.