Automatically designing robot swarms in environments populated by other robots: an experiment in robot shepherding

David Garzón Ramos and Mauro Birattari

Abstract— **Automatic design is a promising approach to realizing robot swarms. Given a mission to be performed by the swarm, an automatic method produces the required control software for the individual robots. Automatic design has concentrated on missions that a swarm can execute independently, interacting only with a static environment and without the involvement of other active entities. In this paper, we investigate the design of robot swarms that perform their mission by interacting with other robots that populate their environment. We frame our research within robot shepherding: the problem of using a small group of robots—the shepherds—to coordinate a relatively larger group—the sheep. In our study, the group of shepherds is the swarm that is automatically designed, and the sheep are pre-programmed robots that populate its environment. We use automatic modular design and neuroevolution to produce the control software for the swarm of shepherds to coordinate the sheep. We show that automatic design can leverage mission-specific interaction strategies to enable an effective coordination between the two groups.**

I. INTRODUCTION

Swarm robotics [1] is an approach to the coordination of large groups of robots [2]. In a robot swarm [2], [3], a collective behavior emerges from the interaction among robots, and between the robots and their environment. The design of collective behaviors for robot swarms is challenging: the mission to be performed is defined at a global level, but the robots must be programmed at the individual level [4]. No generally applicable methodology exists to tell what an individual should do so that a collective behavior emerges in the swarm. Typically, designers of robot swarms rely on manual trial-and-error processes to produce the control software of the robots [5]. Alternatively, this problem has been addressed from the perspective of automatic design [6]–[8]: an optimization process designs the collective behavior of the robots by maximizing a mission-specific performance metric.

In this paper, we study the automatic design of robot swarms that operate in environments populated by other robots. We frame this problem into the robot shepherding problem [9]. In robot shepherding, it is assumed that two groups of robots of different kind operate in the same environment—the shepherds and the sheep. Shepherds and sheep influence each other's behavior and constitute a. heterogeneous system that must perform missions collectively. We use automatic design methods to produce the control software of the shepherds so that they coordinate the sheep in a set of spatially-organizing missions [4], [10]. The sheep operate with pre-defined fixed control software. In a sense, the sheep are reactive robots that populate the environment of the automatically designed swarm of shepherds.

Our goal is to investigate whether an automatic design process can effectively identify and exploit the dynamics between the two groups of robots. In our experiments, the behavior of the sheep is a black-box to the design process. The dynamics of the sheep must be discovered while the automatic design process produces the control software of the shepherds. To investigate this problem, we implement two automatic design methods, `Pistacchio` and `EvoCMY`, and we use them to design the control software of shepherds. We test these two methods in nine experimental scenarios that combine shepherding missions with sheep that operate with diverse predefined control software.

We report qualitative and quantitative results of experiments conducted in simulation. Along with the results obtained with `Pistacchio` and `EvoCMY`, we also report baseline results obtained with a manual design approach and with a simple random walk. The results show that the automatic design methods are effective in identifying and exploiting the dynamics of the scenario to perform the missions at hand. The control software automatically designed performs better than the baselines.

II. RELATED WORK

The traditional approach to the automatic design of robot swarms is neuroevolution [11]–[13]. In this approach, an evolutionary process fine-tunes an artificial neural network that serves as control software for the robots. In the last decade, automatic modular design (AutoMoDe) [14], [15] has been proposed as an alternative to neuroevolution. In the AutoMoDe approach, an optimization process fine-tunes and selects software modules that are assembled into a predefined control architecture. Researchers have investigated various aspects of the automatic design process with the two approaches: design strategies [16]–[18], viable optimization algorithms [19], [20], control architectures [14], [21], robot platforms [22]–[25], emergent collective behaviors [26]–[28], among others. Most swarm robotics research is still conducted under laboratory conditions: mainly, in controlled,

The authors are with IRIDIA, Université libre de Bruxelles, Belgium. {david.garzon.ramos, mauro.birattari}@ulb.be.

The project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (DEMIURGE: grant No. 681872) and from Belgium's Wallonia-Brussels Federation through a ARC Advanced Project 2020 (Guaranteed by Optimization). The authors acknowledge support from Toyota Motor Europe. DGR acknowledges support from Minciencias, Colombia; MB from the Belgian Fonds de la Recherche Scientifique-FNRS.

The authors thank the designers who participated in this study. Implementations and experiments were done by DGR. The paper was drafted by DGR and refined by DGR and MB. The research was directed by MB.

static environments and with homogeneous systems [29]. These scenarios differ from what one could expect in future real deployments of robot swarms [30], [31]—with highly dynamical environments that are populated by robots and other agents [32]. So far, little research has been devoted to investigating how well automatic design can tackle missions that occur in such populated environments.

Robot shepherding has been investigated mostly within the framework of collective motion [33]–[39]. In most cases, the behavior of the sheep is inspired by *flocking* behaviors. Typically, a designer models the desired behavior for the sheep and the shepherd and manually produces control software for the two. The designer uses their knowledge and expertise to tailor the behavior of the shepherds to the behavior of the sheep, which is assumed to be known at design time. Typically, the goal of these studies is to verify whether the models are suitable for creating shepherding behaviors under specific conditions—like with models of other swarm behaviors in the literature [4], [10], [31], [40].

Recent studies have demonstrated that creating such models is a viable principled method to produce specific shepherding behaviors. These studies show that it is possible to coordinate the sheep with a single shepherd [33], [34], [36] or with a group of shepherds that act cooperatively [35], [37]–[39], [41]. Some of these studies have shown the benefits of using optimization processes to fine-tune the parameters of the control software [35], [41]. However, these methods are not conceived to be of general applicability: they are unable to address different types of missions with varied interactions between shepherds and sheep. Indeed, they cannot be directly transferred from one problem to another if the behavior of the sheep varies considerably. For example, a model designed for sheep that move away from the shepherds cannot be ported to problems in which the sheep move towards them.

We show here that automatic design is a general framework for generating shepherding behaviors. In this study, we address varied interactions between shepherds and sheep, without having to apply mission-specific modifications to the design method. We contend that this is a more generally-applicable approach to designing a robot swarm that must interact with other robots in its environment.

III. AUTOMATIC DESIGN METHODS

We produce control software for the shepherds with two automatic design methods: `Pistacchio` and `EvoCMY`. See the supplementary material for an illustrative representation of the methods and the robot [42].

A. Robot platform

`Pistacchio` and `EvoCMY` produce control software for an extended version of the e-puck robot [43], [44]. In this paper, we consider an e-puck whose capabilities are formally defined by the reference model RM 3.3, see Table I. A *reference model*, in this case RM 3.3, defines the interface between the control software and the hardware of the robot. The e-puck defined by RM 3.3 has eight proximity sensors, three ground sensors, an omni-directional vision system, two wheels, and three RGB LEDs. The robot can perceive objects and other robots in a $3\,\mathrm{cm}$ range with its proximity sensors ($prox_i$). It can also detect whether the color of the floor is black, gray, or white with the ground sensors (gnd_j). Using its vision system (cam_c), the robot can perceive peer robots that display the colors cyan, magenta, and yellow with their LEDs. These peers can be perceived in a range of $0.40\,\mathrm{m}$ and with a $360°$ field of view. For each color, there is a vector (V_c) that aggregates the relative position of all the robots perceived. The control software of the robot can set the target velocity (v_k) of the two wheels between -0.12 and $0.12\,\mathrm{m/s}$. The control software can also set the LEDs ($LEDs$) of the robot to display the colors cyan (C), magenta (M), and yellow (Y), or none ($\varnothing$).

TABLE I
REFERENCE MODEL RM 3.3.

Input	Value	Description
$prox_{i\in\{1,\dots,8\}}$	$[0, 1]$	proximity readings
$gnd_{j\in\{1,\dots,3\}}$	$\{black, gray, white\}$	ground readings
$cam_{c\in\{C,M,Y\}}$	$\{yes, no\}$	color signal perceived
$V_{c\in\{C,M,Y\}}$	$\langle 1.0; (0, 2\pi)\,\mathrm{rad}\rangle$	direction of color signal
Output	**Value**	**Description**
$v_{k\in\{l,r\}}$	$(-0.12, 0.12)\,\mathrm{m/s}$	target velocity
$LEDs$	$\{\varnothing, C, M, Y\}$	color signal emitted
Period of the control cycle: $0.1\,\mathrm{s}$		

We conceived RM 3.3 to limit the interactions between shepherds and sheep to those that are triggered by visually perceivable stimuli—i.e., the relative distance between robots and the colors they display. In this way, we simplify the visualization and monitoring of the interactions between the two groups of robots.

`Pistacchio` and `EvoCMY` produce control software that can be transferred between the simulated e-puck and its physical counterpart. In this study, we have only considered the simulated version of the e-puck [44]. We conduct our experiments with ARGoS3 [45]: a swarm robotics simulator that has been used in the past to automatically design collective behaviors that have been subsequently validated in real-robot experiments [14], [19], [23], [27], [46], [47].

B. `Pistacchio`

`Pistacchio` is an automatic design method that belongs to the AutoMoDe family [14], [15]. It fine-tunes, selects, and combines parametric software modules into probabilistic finite-state machines via an optimization process. `Pistacchio`'s modules are adapted from those of `Tutti-Frutti` [23] and enable the e-puck to interact with its peers by perceiving and displaying colors. `Pistacchio` integrates five low-level behavior modules—EXPLORATION, STOP, COLOR-FOLLOWING, COLOR-ELUSION, and CIRCLING—and five transition conditions—BLACK-FLOOR, GRAY-FLOOR, WHITE-FLOOR, FIXED-PROBABILITY, and COLOR-DETECTION. The low-level behaviors are actions that the robot can execute, and the transition conditions

TABLE II

PISTACCHIO'S SOFTWARE MODULES.

Behavior	Parameter	Description
EXPLORATION	$\{\tau, \gamma\}$	random walk with τ rotations
STOP	$\{\gamma\}$	standstill state
COLOR-FOLLOWING	$\{\delta, \gamma\}$	attraction to color δ
COLOR-ELUSION	$\{\delta, \gamma\}$	repulsion from color δ
CIRCLING	$\{\theta, \gamma\}$	circular movement with angle θ
Transition	**Parameter**	**Condition**
BLACK-FLOOR	$\{\beta\}$	detected black floor
GRAY-FLOOR	$\{\beta\}$	detected gray floor
WHITE-FLOOR	$\{\beta\}$	detected white floor
FIXED-PROBABILITY	$\{\beta\}$	fixed probability transition
COLOR-DETECTION	$\{\delta, \beta\}$	detected signal of color δ
All low-level behaviors can also set the LEDs with $\gamma \in \{\varnothing, C, M, Y\}$ β is the probability for a transition to occur if the condition is fulfilled		
`Pistacchio` automatically tunes the parameters τ, θ, δ, γ, and β		

TABLE III

EVOCMY'S NETWORK TOPOLOGY AND EVOLUTION PARAMETERS.

Architecture: Fully-connected feed-forward neural network without hidden layers	
Input nodes	**Description**
$in_{a\in\{1,\ldots,8\}}$	proximity readings $prox_{i\in\{1,\ldots,8\}}$
$in_{a\in\{9,\ldots,11\}}$	ground readings $gnd_{j\in\{1,\ldots,3\}}$
$in_{a\in\{12,\ldots,23\}}$	scalar projections of $V_{c\in\{C,M,Y\}}$
$in_{a\in\{24\}}$	*bias* node
Output nodes	**Mapping**
$out_{b\in\{1,\ldots,4\}}$	velocities $v_{k\in\{l,r\}}$
$out_{b\in\{5,\ldots,8\}}$	color displayed by the LEDs, $\{\varnothing, C, M, Y\}$
Connections	**Description**
$conn_{s\in\{1,\ldots,192\}}$	synaptic weights, with $\omega\in[-5, 5]$
Generations*	computed over the simulations budget
Population size	100
Elitism rate	20
Mutation rate	80
`EvoCMY` automatically tunes the synaptic weights	

are events that can trigger the transition between low-level behaviors. Table II lists `Pistacchio`'s software modules and their parameters.

Given a mission-specific performance metric, `Pistacchio` uses an optimization algorithm to search for configurations of the control software that maximize the performance of the swarm. The control software has the form of a probabilistic-finite state machine with a maximum of four states—the low-level behavior modules—and a maximum of four outgoing transitions per state—the transition modules. A transition always originates and ends in different states. The optimization algorithm used in `Pistacchio` is Iterated F-race [48]—a *de facto* standard optimization algorithm in AutoMoDe methods. Iterated F-race builds finite-states machines and assesses their performance through simulations in ARGoS3. The duration of the optimization process is restricted by a predefined simulation budget. When the simulations budget is exhausted, the design process ends and `Pistacchio` returns the best configuration of control software it found. We test the produced control software in the shepherds without any modification.

C. `EvoCMY`

`EvoCMY` is a straightforward implementation of the neuroevolutionary approach [11]–[13]. It produces artificial neural networks whose synaptic weights are tuned via artificial evolution. `EvoCMY` is an adaptation of `EvoColor` [23]—a method to automatically design collective behaviors for robots that can display and perceive colors. In this sense, like `Pistacchio`, `EvoCMY` produces control software for e-pucks defined by RM 3.3. Table III summarizes the neural network topology and artificial evolution parameters of `EvoCMY`. We do not consider more advanced neuroevolutionary implementations (e.g., `CMA-ES` [49], `xNES` [50], and `NEAT` [51]) as research has shown that they do not provide performance advantages when applied off the shelf [25], [47].

`EvoCMY` uses artificial evolution to search for configurations of the neural network that generate good-performing behaviors. The performance is measured with respect to a mission-specific performance metric that is given as part of the mission specification—the same as for `Pistacchio`. Also, like Iterated F-race in `Pistacchio`, the evolutionary process in `EvoCMY` uses simulations in ARGoS3 to assess the performance of candidate neural networks. The evolutionary process applies elitism and mutation operators to fine-tune the synaptic weights so that the performance of the swarm is optimized. The artificial evolution finishes when a simulations budget is exhausted. Also in this case, we test the produced control software in the shepherds without any modification.

IV. EXPERIMENTAL SETUP

We considered an heterogeneous system of five shepherds and ten sheep, which jointly performed a set of missions. We devised the control software for the sheep so that they did not take action unless stimulated by the shepherds. In this way, the performance of the heterogeneous system strictly depended on the effectiveness of the shepherding behaviors that were designed.

A. Sheep control software

Each sheep operated with one out of three pre-defined instances of control software: *C1-Attraction*, *C2-Repulsion*, and *C3-Attraction&Repulsion*. Shepherds could stimulate the sheep by physical proximity in a 3 cm range, or by displaying colors with their LEDs in a 40 cm range. The ground sensor of the sheep allowed them to detect regions of interest in the environment.

In *C1-Attraction*, the sheep were attracted to shepherds that display the color magenta. In *C2-Repulsion*, the sheep were repelled from shepherds that display the color cyan. In *C3-Attraction&Repulsion*, the sheep were both attracted to shepherds that display the color magenta and repelled from

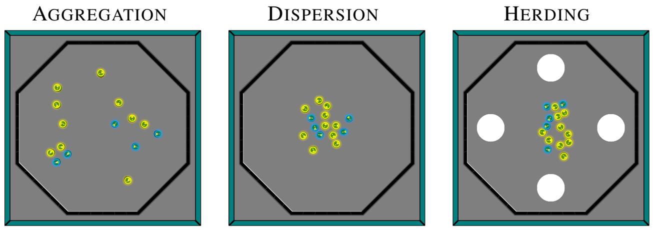

Fig. 1. Experimental arenas for AGGREGATION, DISPERSION, and HERDING. The figure shows an example of the starting positions of five shepherds (cyan) and ten sheep (yellow) in each mission.

shepherds that display the color cyan. In all cases, the sheep displayed the color yellow and remained static if no stimuli were perceived—physical proximity or color. If the sheep stepped into a white-floor region, it halted its movement and turned off its LEDs until the end of the mission.

C1-Attraction, *C2-Repulsion*, and *C3-Attraction&Repulsion* are probabilistic finite-state machines created with software modules that are similar to those of `Pistacchio`—see supplementary material. We followed a manual trial-and-error process to design them. These finite-state machines were undisclosed to the design process that generates the behavior of the shepherds, which sees them as a black box.

B. Missions

The robots operated in an octagonal arena of about $2.8\,\mathrm{m}^2$ and gray floor. The arena had white floor regions on a per-mission basis—see Fig. 1. The robots had a time $T = 120\,\mathrm{s}$ to perform a mission.

1) AGGREGATION*:* Shepherds and sheep are randomly distributed at the arena. The shepherds must group the sheep. The objective function $F_1 = \frac{\sum_{i=1}^{10} D_i(T)}{10}$, which must be minimized, is the average distance from each sheep to the center of mass of all sheep at the end of the mission. $D_i(T)$ is the distance from a sheep i at position (x_i, y_i) to the center of mass of all sheep (x_c, y_c) at time T.

2) DISPERSION*:* Shepherds and sheep are randomly distributed at the center of the arena. The shepherds must separate the sheep. The objective function F_1, which in this case must be maximized, measures the average distance from each sheep to the center of mass of all sheep at the end of the mission.

3) HERDING*:* Shepherds and sheep are randomly distributed at the center of the arena. The shepherds must drive the sheep to four indicated locations. The indicated locations are white circular regions of about $0.3\,\mathrm{m}^2$. The four locations are equivalent to each other. The objective function $F_2 = K_\mathrm{n}(T)$, which must be minimized, is the number of sheep that remain out of the four locations at the end of the mission, at time T.

By pairing sheep control software and missions, we presented varied challenges to the design of the shepherding behaviors. The sheep could be more or less cooperative with the shepherds for the mission at hand. For example, we expected the shepherds to group the sheep more effectively when the sheep operated with *C1-Attraction*, and less effectively when they operated with *C2-Repulsion*. Conversely, we expected the shepherds to separate the sheep more effectively when the sheep operated with *C2-Repulsion*, and less effectively when they operated with *C1-Attraction*. *C3-Attraction&Repulsion* gives more freedom to the automatic design process to select the best performing strategy.

C. Baseline

We also report results obtained with manual design and a random walk. These results are a reference baseline to appraise the performance of the automatic methods.

1) `C-Human`*:* a manual design method introduced by Francesca *et al.* [52] to appraise the performance of automatic modular design methods. In `C-Human`, a group of human designers fine-tune and combine parametric software modules to produce control software for the robots [19], [21], [52]. In our experiments, they produced control software for the shepherds. We invited three designers with more than one year of experience in swarm robotics and some familiarity with AutoMoDe, the e-puck, and ARGoS3. The designers were provided with `Pistacchio`'s software modules and they were restricted to produce control software with its control architecture. In this sense, the designers adopted the role of an optimization algorithm. They searched the space of possible control software to generate the desired shepherding behaviors.

2) `R-Walk`*:* a trivial implementation of a random walk behavior in which the shepherds move with a ballistic motion [53]. `R-Walk` is not a design method, as no optimization process is conducted. We include it as a lower bound to the performance, as in recent automatic design studies [24], [47].

D. Protocol

We conducted experiments that pair the three missions and the three instances of sheep control software. The pairing resulted in nine experimental scenarios. Neither the automatic methods nor the human designers had direct access to information about the sheep control software. The dynamics between shepherds and sheep had to be discovered during the design process via simulations. The experimental protocol we followed is based on previous studies on the automatic design of robot swarms [19], [21], [52]. `Pistacchio` and `EvoCMY` were given a budget of 100 000 simulations to produce every instance of control software. Producing control software with `Pistacchio` and `EvoCMY` costs less than producing it with `C-Human`. Therefore, we used them to repeatedly produce more instances of control software than what `C-Human` could produce. The number of evaluations was adjusted to obtain an equivalent number of observations for statistical analysis. We produced 90 instances of control software with `Pistacchio` and other 90 with `EvoCMY`—10 per scenario. Each of these instances was assessed once to obtain 90 observations per method. `C-Human` produced 9 instances of control software, 1 per scenario. We obtained the equivalent 90 observations by assessing each of these instances 10 times. `R-Walk` was assessed 10 times in each scenario to obtain 90 observations.

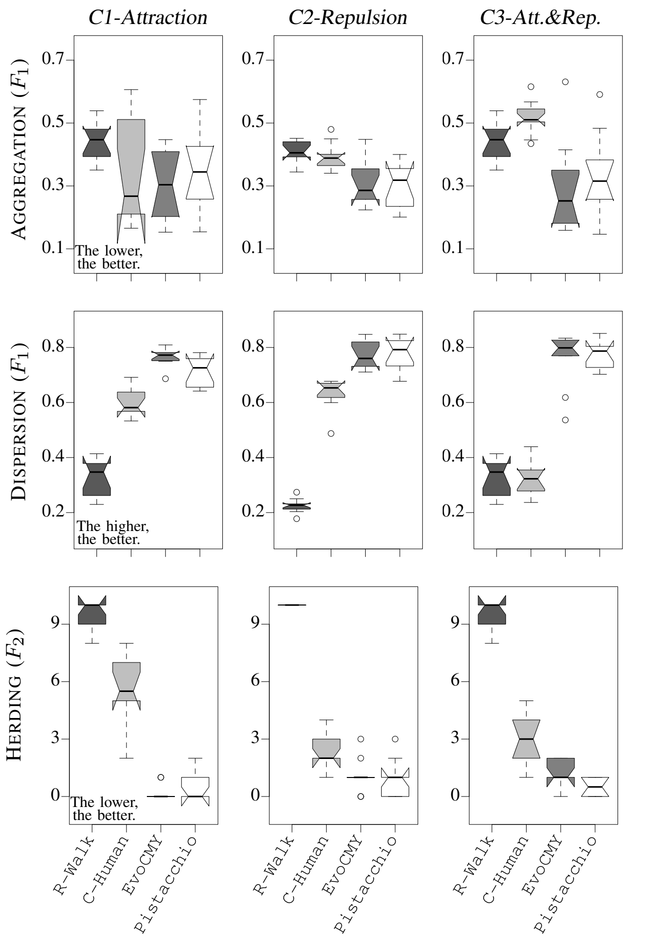

Fig. 2. Results per mission and sheep control software. The plots show the score obtained in the nine experimental scenarios, ten observations per method and scenario. Results per mission are organized in rows. Results per sheep control software are organized in columns. Results per design method are presented with grayscale box-plots, `R-Walk` (■), `C-Human` (■), `EvoCMY` (■), `Pistacchio` (□).

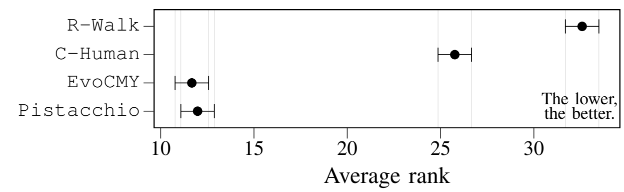

Fig. 3. Friedman test that aggregates the results obtained in the nine experimental scenarios. The plot shows the average rank of each method and its confidence interval.

We report numerical results with notched box-plots. Additionally, we use a Friedman test to report aggregate results across the nine experimental scenarios. Comparisons between methods are statistically supported by the computation of the 95 % confidence interval [54]. The performance of two methods is considered statistically different if the confidence intervals of their median do not overlap.

V. EXPERIMENTAL RESULTS

The control software, collected data, and demonstrative videos are available as supplementary material [42]. Fig. 2 shows box-plots of the score obtained in each mission. Fig. 3 shows the average rank of the design methods over the nine experimental scenarios. The per-mission results and the Friedman test did not detect any significant difference between the performance of `Pistacchio` and `EvoCMY`, but the two are significantly better than `C-Human` and `R-Walk`. Moreover, `C-Human` was significantly better than `R-Walk`. The results show that automatic design was more effective than manual design in addressing the shepherding problems we considered. Also, all design methods generated collective behaviors that are more effective than the simple random walk—the lower bound. Our simulation-only comparison between `Pistacchio` and `EvoCMY` was not sufficient to identify possible performance differences between the modular and the neuroevolutionary approach—which have been previously reported in similar studies [19], [55].

A. Shepherding strategies

`Pistacchio` and `EvoCMY` leveraged coordination and localization via color signals to create effective interactions between shepherds and sheep. In most cases, the two methods designed behaviors in which the shepherds stimulated the sheep in similar ways. We describe below some shepherding strategies that were generated. The following are general observations we made over all instances of control software.

1) Grouping the sheep in AGGREGATION*:* When the sheep operated with *C1-Attraction*, the shepherds displayed magenta to attract them and they were themselves also attracted to other shepherds that displayed magenta. In this way, shepherds and sheep remained close to each other—keeping the sheep close to their center of mass. The automatic methods designed a coordinated cooperative behavior between shepherds. When the sheep operated with a *C2-Repulsion*, the shepherds displayed cyan and moved on a circular trajectory close to the walls of the arena. The sheep were therefore steadily repelled towards the center of the arena and formed a single group. When the sheep operated with *C3-Attraction&Repulsion*, the shepherds used a behavior similar to that observed in *C1-Attraction* or *C2-Repulsion*—no noticeable preference was observed.

2) Separating the sheep in DISPERSION*:* When the sheep operated with *C1-Attraction*, the shepherds remained close to the walls and displayed magenta to attract the sheep. This behavior dispersed the sheep along the edges of the arena—keeping them far from their center of mass. When the sheep operated with *C2-Repulsion*, the shepherds moved in circles in the center of the arena while displaying cyan. In this way, the shepherds separated the sheep by steadily pushing them towards the walls. When the sheep operated with *C3-Attraction&Repulsion*, the shepherds used a behavior similar to that observed in *C1-Attraction* or *C2-Repulsion*—also no noticeable preference was observed.

3) Driving the sheep in HERDING*:* The shepherds reacted to the color yellow that the sheep displayed. When the sheep operated with *C1-Attraction*, the shepherds displayed magenta to attract the sheep. Simultaneously, they were also repelled from the color yellow the sheep displayed. In this way, the shepherds guided the sheep from the front. The two navigated the arena together until the sheep stepped into a white region and turned off their LEDs. When the sheep operated with *C2-Repulsion*, the shepherds displayed cyan to repel the sheep. Unlike the behavior observed in *C1-Attraction*, in this case, the shepherds were attracted to the color yellow that the sheep displayed. The simultaneous execution of these two behaviors resulted in shepherds that chased the sheep until the latter stepped into a white region and turned off their LEDs. The shepherds used a similar behavior to that observed in *C1-Attraction* or *C2-Repulsion* when the sheep operated with *C3-Attraction&Repulsion*. Also in this case, no noticeable preference was observed.

Color signaling and physical proximity were two alternative ways for the shepherds to interact with the sheep. Unlike `Pistacchio` and `EvoCMY`, `C-Human` produced control software that leveraged the color displayed by the shepherds only in a few cases. `C-Human` and `Pistacchio` operate on the same set of modules and software architecture, and can potentially produce control software that performs similarly. However, the human designers mainly used sub-optimal strategies (based on physical proximity) to enable the interaction between shepherds and sheep. In a way, they failed to discover and use the most effective strategy: the interaction via colors.

VI. DISCUSSION

The automatic methods effectively searched the design space and exploited the dynamics between shepherd and sheep. The shepherds used color stimuli to interact with the sheep in a meaningful and good-performing way. In some cases, the shepherds coordinated with each other using the same stimuli. Past research has shown that automatic design and simple color signaling are sufficient to generate mission-specific coordination and spatial-organization strategies [23], [24]. These results were obtained with a homogeneous robot swarm. In our experiments, `Pistacchio` and `EvoCMY` generated similar mission-specific coordination and spatial organization between shepherds and sheep. We show therefore that this ability of automatic methods extends to heterogeneous setups.

So far, automatic design research has focused mainly on missions performed by a single robot swarm that operates alone. With our experiments, we showed that existing approaches are also well suited to address more complex heterogeneous scenarios. We consider heterogeneity in both the number of swarms that populate an environment and the type of control software they use. In our experiments, the sheep operated with finite-state machines. `Pistacchio` and `EvoCMY` generated effective shepherding strategies despite that their shepherds operated with different kinds of control software—finite-state machines and neural networks, respectively. A key element to enable a setup with heterogeneous control software was to formally define the shepherd and sheep robot capabilities within a single reference model, RM 3.3.

A priori, we expected to observe a significant performance difference in AGGREGATION and DISPERSION when the sheep operated with diametrically different behaviors such as *C1-Attraction* and *C2-Repulsion*. However, no such performance difference was observed. This indicates that the automatic methods simultaneously tailored the control software of the shepherds to the one of the sheep, and to the goal of the mission—regardless of the combination. On the other hand, `C-Human` was less effective. Human designers had difficulties on exploring the design space and finding good-performing shepherding strategies. We conceived the experiments in a way that the dynamics between shepherd and sheep had to be discovered during the design process. Automatic methods have a notable advantage in this task. The optimization process is more effective than a human designer in exploring the large and complex design space.

The shepherding problem was an appropriate framework to study the design of robot swarms that must interact with other robots. Our current experimental setup can be directly extended to missions that involve other types of interactions. The sheep we considered are rather individualistic: they react individually to the stimuli of the shepherds without considering the behavior of other sheep (beyond the physical proximity). Moreover, their naturally-static behavior made them easy to handle for shepherds. One could possibly create more complex missions with sheep that continuously move and operate with a more coordinated collective behavior—both cooperative or adversarial. In this sense, our research could bootstrap recent studies on the design of robot swarms that are robust to attacks from adversarial robots [56]–[59].

VII. CONCLUSION

Automatic design is a viable approach to producing swarms that operate in environments populated by other robots. We showed this with experiments conducted in a robot shepherding problem. The automatic methods designed control software for the shepherds, which effectively coordinated groups of pre-programmed sheep. The automatic methods leveraged mission-specific coordination and spatial organization to generate good-performing control software. These capabilities were shown in the past in the automatic design of homogeneous robot swarms. We showed they also hold for an heterogeneous system that must operate collectively—the shepherds and sheep.

In our experiments, the sheep operated with finite-state machines and the shepherds operated either with finite-state machines or artificial neural networks. In this sense, we demonstrated the automatic design of collective behaviors between swarms that operate with different control architecture. We did so by experimenting with the modular and neuroevolutionary approaches. We expect these experiments can motivate new research on the automatic design of heterogeneous robot swarms—with multiple different robot swarms designed at once. We will further investigate this idea with the simultaneous design of shepherds and sheep.

REFERENCES

[1] H. Hamann, *Swarm robotics: a formal approach*. Cham, Switzerland: Springer, 2018.

[2] E. Şahin, "Swarm robotics: from sources of inspiration to domains of application," in *Swarm Robotics: SAB 2004 International Workshop*, ser. Lecture Notes in Computer Science, E. Şahin and W. M. Spears, Eds., vol. 3342. Berlin, Germany: Springer, 2005, pp. 10–20.

[3] G. Beni, "From swarm intelligence to swarm robotics," in *Swarm Robotics: SAB 2004 International Workshop*, ser. Lecture Notes in Computer Science, E. Şahin and W. M. Spears, Eds., vol. 3342. Berlin, Germany: Springer, 2005, pp. 1–9.

[4] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo, "Swarm robotics: a review from the swarm engineering perspective," *Swarm Intelligence*, vol. 7, no. 1, pp. 1–41, 2013.

[5] D. St-Onge, V. S. Varadharajan, Š. Ivan, and G. Beltrame, "From design to deployment: decentralized coordination of heterogeneous robotic teams," *Frontiers in Robotics and AI*, vol. 7, p. 51, 2020.

[6] M. Birattari, A. Ligot, D. Bozhinoski, M. Brambilla, G. Francesca, L. Garattoni, D. Garzón Ramos, K. Hasselmann, M. Kegeleirs, J. Kuckling, F. Pagnozzi, A. Roli, M. Salman, and T. Stützle, "Automatic off-line design of robot swarms: a manifesto," *Frontiers in Robotics and AI*, vol. 6, p. 59, 2019.

[7] M. Birattari, A. Ligot, and K. Hasselmann, "Disentangling automatic and semi-automatic approaches to the optimization-based design of control software for robot swarms," *Nature Machine Intelligence*, vol. 2, no. 9, pp. 494–499, 2020.

[8] G. Francesca and M. Birattari, "Automatic design of robot swarms: achievements and challenges," *Frontiers in Robotics and AI*, vol. 3, no. 29, pp. 1–9, 2016.

[9] J.-M. Lien, O. B. Bayazit, R. T. Sowell, S. Rodríguez, and N. M. Amato, "Shepherding behaviors," in *2004 IEEE International Conference on Robotics and Automation (ICRA)*, vol. 4. Piscataway, NJ, USA: IEEE, 2004, pp. 4159–4164.

[10] M. Schranz, M. Umlauft, M. Sende, and W. Elmenreich, "Swarm robotic behaviors and current applications," *Frontiers in Robotics and AI*, vol. 7, p. 36, 2020.

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

[12] V. Trianni, *Evolutionary Swarm Robotics*. Berlin, Germany: Springer, 2008.

[13] S. Nolfi, *Behavioral and Cognitive Robotics: An Adaptive Perspective*. Rome, Italy: Institute of Cognitive Sciences and Technologies, National Research Council, 2021.

[14] G. Francesca, M. Brambilla, A. Brutschy, V. Trianni, and M. Birattari, "AutoMoDe: a novel approach to the automatic design of control software for robot swarms," *Swarm Intelligence*, vol. 8, no. 2, pp. 89–112, 2014.

[15] M. Birattari, A. Ligot, and G. Francesca, "AutoMoDe: a modular approach to the automatic off-line design and fine-tuning of control software for robot swarms," in *Automated Design of Machine Learning and Search Algorithms*, ser. Natural Computing Series, N. Pillay and R. Qu, Eds. Cham, Switzerland: Springer, 2021, pp. 73–90.

[16] A. L. Christensen and M. Dorigo, "Evolving an integrated phototaxis and hole-avoidance behavior for a swarm-bot," in *Artificial Life X: Proceedings of the Tenth International Conference on the Simulation and Synthesis of Living Systems*, L. M. Rocha, L. S. Yaeger, M. A. Bedau, D. Floreano, R. L. Goldstone, and A. Vespignani, Eds. Cambridge, MA, USA: MIT Press, 2006, pp. 248–254, a Bradford Book.

[17] S. Jones, A. Winfield, S. Hauert, and M. Studley, "Onboard evolution of understandable swarm behaviors," *Advanced Intelligent Systems*, vol. 1, no. 6, p. 1900031, 2019.

[18] I. Gharbi, J. Kuckling, D. Garzón Ramos, and M. Birattari, "Show me what you want: inverse reinforcement learning to automatically design robot swarms by demonstration," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. Piscataway, NJ, USA: IEEE, 2023, pp. 5063–5070.

[19] G. Francesca, M. Brambilla, A. Brutschy, L. Garattoni, R. Miletitch, G. Podevijn, A. Reina, T. Soleymani, M. Salvaro, C. Pinciroli, F. Mascia, V. Trianni, and M. Birattari, "AutoMoDe-Chocolate: automatic design of control software for robot swarms," *Swarm Intelligence*, vol. 9, no. 2–3, pp. 125–152, 2015.

[20] N. Cambier and E. Ferrante, "AutoMoDe-Pomodoro: an evolutionary class of modular designs," in *GECCO'22: Proceedings of the Genetic and Evolutionary Computation Conference*, J. E. Fieldsend, Ed. New York, NY, USA: ACM, 2022, pp. 100–103.

[21] J. Kuckling, V. van Pelt, and M. Birattari, "Automatic modular design of behavior trees for robot swarms with communication capabilities," in *Applications of Evolutionary Computation: 24th International Conference, EvoApplications 2021*, ser. Lecture Notes in Computer Science, P. A. Castillo and J. L. Jiménez Laredo, Eds., vol. 12694. Cham, Switzerland: Springer, 2021, pp. 130–145.

[22] M. Duarte, V. Costa, J. Gomes, T. Rodrigues, F. Silva, S. M. Oliveira, and A. L. Christensen, "Evolution of collective behaviors for a real swarm of aquatic surface robots," *PLOS ONE*, vol. 11, no. 3, p. e0151834, 2016.

[23] D. Garzón Ramos and M. Birattari, "Automatic design of collective behaviors for robots that can display and perceive colors," *Applied Sciences*, vol. 10, no. 13, p. 4654, 2020.

[24] M. Salman, D. Garzón Ramos, and M. Birattari, "Automatic design of stigmergy-based behaviours for robot swarms," *Communications Engineering*, vol. 3, p. 30, 2024.

[25] M. Kegeleirs, D. Garzón Ramos, K. Hasselmann, L. Garattoni, G. Francesca, and M. Birattari, "Transferability in the automatic off-line design of robot swarms: from sim-to-real to embodiment and design-method transfer across different platforms," *IEEE Robotics and Automation Letters*, vol. 9, no. 3, pp. 2758–2765, 2024.

[26] V. Trianni, R. Groß, H. Labella, Thomas, E. Şahin, and M. Dorigo, "Evolving aggregation behaviors in a swarm of robots," in *Advances in Artificial Life: 7th European Conference, ECAL 2003*, ser. Lecture Notes in Computer Science, W. Banzhaf, J. Ziegler, T. Christaller, P. Dittrich, and J. T. Kim, Eds., vol. 2801. Berlin, Germany: Springer, 2003, pp. 865–874.

[27] E. Ferrante, A. E. Turgut, E. A. Duéñez-Guzmán, M. Dorigo, and T. Wenseleers, "Evolution of self-organized task specialization in robot swarms," *PLOS Computational Biology*, vol. 11, no. 8, p. e1004273, 2015.

[28] F. J. Mendiburu, D. Garzón Ramos, M. R. A. Morais, A. M. N. Lima, and M. Birattari, "AutoMoDe-Mate: automatic off-line design of spatially-organizing behaviors for robot swarms," *Swarm and Evolutionary Computation*, vol. 74, p. 101118, 2022.

[29] H. Hamann, M. Schranz, W. Elmenreich, V. Trianni, C. Pinciroli, N. Bredeche, and E. Ferrante, "Editorial: designing self-organization in the physical realm," *Frontiers in Robotics and AI*, vol. 7, p. 164, 2020.

[30] M. Dorigo, G. Theraulaz, and V. Trianni, "Reflections on the future of swarm robotics," *Science Robotics*, vol. 5, p. eabe4385, 2020.

[31] ——, "Swarm robotics: past, present, and future [point of view]," *Proceedings of the IEEE*, vol. 109, no. 7, pp. 1152–1165, 2021.

[32] A. J. King, S. J. Portugal, D. Strömbom, R. P. Mann, J. A. Carrillo, D. Kalise, G. de Croon, H. Barnett, P. Scerri, R. Groß, D. R. Chadwick, and M. Papadopoulou, "Biologically inspired herding of animal groups by robots," *Methods in Ecology and Evolution*, vol. 14, no. 2, pp. 478–486, 2023.

[33] K. Genter and P. Stone, "Influencing a flock via ad hoc teamwork," in *Swarm Intelligence: 9th International Conference, ANTS 2014*, ser. Lecture Notes in Computer Science, M. Dorigo, M. Birattari, S. Garnier, H. Hamann, M. Montes de Oca, C. Solnon, and T. Stützle, Eds., vol. 8667. Cham, Switzerland: Springer International Publishing, 2014, pp. 110–121.

[34] ——, "Adding influencing agents to a flock," in *AAMAS '16: Proceedings of the 2016 International Conference on Autonomous Agents and Multiagent Systems*. Richland, SC, USA: International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS), 2016, pp. 615–623.

[35] A. Özdemir, M. Gauci, and R. Groß, "Shepherding with robots that do not compute," in *ECAL 2017, the Fourteenth European Conference on Artificial Life*. Cambridge, MA, USA: MIT Press, 2017, pp. 332–339.

[36] R. A. Licitra, Z. I. Bell, and W. E. Dixon, "Single-agent indirect herding of multiple targets with uncertain dynamics," *IEEE Transactions on Robotics*, vol. 35, no. 4, pp. 847–860, 2019.

[37] A. Pierson and M. Schwager, "Controlling noncooperative herds with robotic herders," *IEEE Transactions on Robotics*, vol. 34, no. 2, pp. 517–525, 2018.

[38] J. Hu, A. E. Turgut, T. Krajník, B. Lennox, and F. Arvin, "Occlusion-based coordination protocol design for autonomous robotic shepherding tasks," *IEEE Transactions on Cognitive and Developmental Systems*, p. 1, 2020.

[39] E. Sebastián, E. Montijano, and C. Sagüés, "Adaptive multirobot implicit control of heterogeneous herds," *IEEE Transactions on Robotics*, vol. 38, no. 6, pp. 3622–3635, 2022.

[40] N. Nedjah and L. Silva Junior, "Review of methodologies and tasks in swarm robotics towards standardization," *Swarm and Evolutionary Computation*, vol. 50, p. 100565, 2019.

[41] G. Y. Dosieah, A. Özdemir, M. Gauci, and R. Groß, "Moving mixtures of active and passive elements with robots that do not compute," in *Swarm Intelligence: 13th International Conference, ANTS 2022*, ser. Lecture Notes in Computer Science, vol. 13491. Cham, Switzerland: Springer, 2022, pp. 183–195.

[42] D. Garzón Ramos and M. Birattari, "Automatically designing robot swarms in environments populated by other robots, an experiment in robot shepherding: supplementary material," https://iridia.ulb.ac.be/supp/IridiaSupp2023-002, 2024.

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

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

[45] C. Pinciroli, V. Trianni, R. O'Grady, G. Pini, A. Brutschy, M. Brambilla, N. Mathews, E. Ferrante, G. A. Di Caro, F. Ducatelle, M. Birattari, L. M. Gambardella, and M. Dorigo, "ARGoS: a modular, parallel, multi-engine simulator for multi-robot systems," *Swarm Intelligence*, vol. 6, no. 4, pp. 271–295, 2012.

[46] V. Trianni and M. López-Ibáñez, "Advantages of task-specific multi-objective optimisation in evolutionary robotics," *PLOS ONE*, vol. 10, no. 8, p. e0136406, 2015.

[47] K. Hasselmann, A. Ligot, J. Ruddick, and M. Birattari, "Empirical assessment and comparison of neuro-evolutionary methods for the automatic off-line design of robot swarms," *Nature Communications*, vol. 12, p. 4345, 2021.

[48] M. López-Ibáñez, J. Dubois-Lacoste, L. Pérez Cáceres, M. Birattari, and T. Stützle, "The irace package: iterated racing for automatic algorithm configuration," *Operations Research Perspectives*, vol. 3, pp. 43–58, 2016.

[49] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evolutionary Computation*, vol. 9, no. 2, pp. 159–195, 2001.

[50] T. Glasmachers, T. Schaul, S. Yi, D. Wierstra, and J. Schmidhuber, "Exponential natural evolution strategies," in *GECCO'10: Proceedings of the 12th annual conference on Genetic and evolutionary computation*. New York, NY, USA: ACM, 2010, pp. 393–400.

[51] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary Computation*, vol. 10, no. 2, pp. 99–127, 2002.

[52] G. Francesca, M. Brambilla, A. Brutschy, L. Garattoni, R. Miletitch, G. Podevijn, A. Reina, T. Soleymani, M. Salvaro, C. Pinciroli, V. Trianni, and M. Birattari, "An experiment in automatic design of robot swarms: AutoMoDe-Vanilla, EvoStick, and human experts," in *Swarm Intelligence: 9th International Conference, ANTS 2014*, ser. Lecture Notes in Computer Science, M. Dorigo, M. Birattari, S. Garnier, H. Hamann, M. Montes de Oca, C. Solnon, and T. Stützle, Eds., vol. 8667. Cham, Switzerland: Springer International Publishing, 2014, pp. 25–37.

[53] M. Kegeleirs, D. Garzón Ramos, and M. Birattari, "Random walk exploration for swarm mapping," in *Towards Autonomous Robotic Systems: 20th Annual Conference, TAROS 2019*, ser. Lecture Notes in Computer Science, K. Althoefer, J. Konstantinova, and K. Zhang, Eds., vol. 11650. Cham, Switzerland: Springer, 2019, pp. 211–222.

[54] W. J. Conover, *Practical Nonparametric Statistics*, 3rd ed., ser. Wiley Series in Probability and Statistics. New York, NY, USA: John Wiley & Sons, 1999.

[55] A. Ligot and M. Birattari, "Simulation-only experiments to mimic the effects of the reality gap in the automatic design of robot swarms," *Swarm Intelligence*, vol. 14, pp. 1–24, 2020.

[56] V. Strobel, E. Castelló Ferrer, and M. Dorigo, "Blockchain technology secures robot swarms: a comparison of consensus protocols and their resilience to byzantine robots," *Frontiers in Robotics and AI*, vol. 7, p. 54, 2020.

[57] E. Castelló Ferrer, T. Hardjono, A. Pentland, and M. Dorigo, "Secure and secret cooperation in robot swarms," *Science Robotics*, vol. 6, no. 56, p. eabf1538, 2023.

[58] V. Strobel, A. Pachecho, and M. Dorigo, "Robot swarms neutralize harmful byzantine robots using a blockchain-based token economy," *Science Robotics*, vol. 8, no. 79, p. eabm4636, 2023.

[59] A. Reina, R. Zakir, G. De Masi, and E. Ferrante, "Cross-inhibition leads to group consensus despite the presence of strongly opinionated minorities and asocial behaviour," *Communications Physics*, vol. 6, p. 236, 2023.