2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Automatic design of robot swarms that perform composite missions: an approach based on inverse reinforcement learning

Jeanne Szpirer, David Garzón Ramos, and Mauro Birattari

***Abstract*— We investigate the automatic design of robot swarms that perform composite missions—that is, missions specified as the composition of consecutive sub-missions. Automatic design through performance optimization has become a viable and appealing approach to designing robot swarms. First, a user defines a mission by specifying a performance measure: a function indicating to what extent the swarm has attained its goal. An optimization process then generates suitable control software for the robots by maximizing the performance measure. The definition of a performance measure is a challenging task that requires expert input, which hinders the automatic nature of the approach. Recently, inverse reinforcement learning was introduced to minimize the need for human intervention in the automatic design of robot swarms. However, this method was only applied to single-objective missions. In this paper, we extend the method to address composite missions, by formulating and solving the design problem as a multi-objective optimization problem. We conduct simulations with a swarm of twenty e-puck robots that perform twelve composite missions. We compare the performance of the swarm when the robots operate with control software produced manually or using inverse reinforcement learning.**

I. INTRODUCTION

Swarm robotics [1]–[3] has gained notable attention in the past decade [4]–[12]. However, designing robot swarms is still a major challenge [13]: no generally applicable method exists to generate the control software of individual robots so that they can collectively perform a given mission [14]–[16]. A promising approach to address this issue is automatic design [17]–[28]. In automatic design, the control software is generated via an optimization-based process [29]. This process is typically driven by a mission-specific performance measure—i.e., an objective function—provided as part of the mission specification. During the design process, an optimization algorithm searches the space of possible instances of control software to find one that maximizes the performance of the swarm—according to the mission specification. No human intervention is required in the realization of robot swarms, except for the definition of the mission specification, including the performance measure. The problem is that formulating a performance measure that properly specifies the desired behavior of a swarm is a non-trivial task that requires the attention of an expert [30], [31].

We contend that automatic design must advance towards methods that operate over a more natural and intuitive form of mission specification—e.g., simplified language [32] or behavior demonstrations [33], [34]. This is a necessary step to achieve truly automatic design processes that allow for the design and deployment of robot swarms without requiring technical expertise [21].

The challenge of formulating appropriate performance measures has emerged also within the reinforcement learning literature, particularly in the context of developing a suitable reward function that facilitates the learning of an optimal policy [35]. Inverse reinforcement learning was conceived to tackle this problem [36]. Instead of manually crafting a reward function, inverse reinforcement learning methods automatically generate a reward function from a user's demonstrations. The reward function and the policy are learned simultaneously in an iterative process. This approach is advantageous because demonstrating desired behaviors is typically easier, more natural, and intuitive, than crafting a corresponding reward function [37], [38]. These advantages recently motivated the adoption of inverse reinforcement learning in the automatic design of robot swarms [33], [34].

In this paper, we use inverse reinforcement learning to automatically design robot swarms that perform composite missions. The composite missions are specified by two sub-missions that must be performed one after the other. A cue in the environment signals to the swarm which sub-mission to perform at every moment. The two sub-missions are specified via demonstrations of the desired spatial organization that the swarm must achieve. The challenges of addressing this design problem are (i) to identify relevant features that characterize the sub-missions using only demonstrations and (ii) to concurrently consider two sub-missions in the design of a single instance of control software.

In our study, we consider two design approaches with different degrees of human intervention: a multi-objective approach (DemoTuttiFrutti-`MO`) and a two single-objective one (DemoTuttiFrutti-`2SO`). For simplicity, we will refer to these methods as DTF-`MO` and DTF-`2SO`, respectively, throughout the paper. In DTF-`MO`, the sub-missions are addressed concurrently and the design process generates a single instance of control software from the demonstrations, without any human intervention. In DTF-`2SO`, the sub-missions are addressed independently and the design process generates two instances of control software—one for each demonstrated sub-mission. Then, we manually hard code a

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

The project has received funding from Belgium's Wallonia-Brussels Federation through a ARC Advanced Project 2020 (Guaranteed by Optimization). DGR acknowledges support from the Colombian Ministry of Science, Technology and Innovation – Minciencias; and JS and MB from the Belgian Fonds de la Recherche Scientifique–FNRS.

The experiments were designed by JS and DGR, and performed by JS. The paper was drafted by JS and revised by JS, DGR, and MB. The research was directed by MB.

transition rule that allows the robots to switch from one sub-mission to the other [39]. We compare the results of these two approaches with those obtained by applying a baseline—control software produced completely by hand.

We report qualitative and quantitative results of experiments conducted with a swarm of twenty e-puck robots [40] in ARGoS3 [41]—a realistic physics-based swarm robotics simulator that proved to be able to produce control software that transfers effectively to the real robots [12], [42]–[44]. For most missions, DTF-2SO and/or DTF-MO performed better than the manual baseline, with DTF-2SO often performing better than DTF-MO. In DTF-MO, the design process was able to generate tailored transition rules that allowed the swarm to switch from one behavior to the other. In particular, it found strategies that exploit the communication capabilities of the robots to coordinate the transition. However, this fully automatic approach considerably increased the size of the search space with respect to the semi-automatic one, which ultimately hindered its relative performance. The experiments we conducted allowed us to identify the reasons behind this performance difference and to outline steps to overcome this challenge in future research.

II. RELATED WORK

In swarm robotics, inverse reinforcement learning was first applied by Šošic *et al.* [33]. They presented a scalable solution to learning a local reward function that explains and reproduces the desired global behavior of a swarm. In their paper, Šošic *et al.* designed collective behaviors for a swarm of particles that perform collective motion and alignment. More recently, Gharbi *et al.* [34] built on these ideas and applied inverse reinforcement learning to the automatic design of control software for a swarm of twenty e-puck robots. They introduced Demo-Cho: a method to realize robot swarms starting from user demonstrations. Demo-Cho is the combination of Chocolate [25]—a modular method to automatically design robot swarms—and apprenticeship learning [36]—an implementation of the inverse reinforcement learning approach. In their paper, Gharbi *et al.* focused on proving the general applicability of inverse reinforcement learning to automatically designing robot swarms in several missions. The authors restricted their study to missions aimed at achieving a single desired behavior [11]. Like in Šošic's work, Gharbi *et al.* conducted experiments with a swarm that must achieve a desired spatial organization in its environment. For a recent review on robot learning for swarm robotics, see [45].

Little research exists on how to automatically design robot swarms that must perform composite missions, transitioning from one sub-mission to another. Previous related studies [46]–[48] assumed that the sub-missions are learned separately, and the transitions and/or order are known beforehand. Notably, Duarte *et al.* [39] evolved individual neural network-based controllers to achieve desired behaviors. Subsequently, they combined these to generate control software capable of addressing composite missions. On the other hand, Garattoni and Birattari [6] presented a swarm that is able to collectively sequence sub-missions at run time, without the need to know a priori the order in which they should be executed. However, the behaviors needed to perform the sub-missions were hard-coded in advance.

Designing robot swarms that perform composite missions is more complex than designing them to perform a single mission at a time. A user must produce control software that performs well in all the specified sub-missions, while enabling efficient transitions from one to another [19], [49], [50]. In its own nature, this is a multi-criteria design problem [51]. Šošic's and Gharbi's approaches cannot be directly applied to problems that belong into a multi-criteria design framework. The methods they conceived do not provide the means for (i) specifying various missions altogether and (ii) conducting the required multi-criteria optimization process. In our research, we address these two limitations and investigate how to address composite missions in an automatic and integrated way. By doing so, we advance the automatic design of robot swarms towards missions of growing complexity, while simultaneously reducing the technical expertise required to realize them.

III. DESIGNING ROBOT SWARMS THAT PERFORM COMPOSITE MISSIONS

The approaches we propose—DTF-MO and DTF-2SO—belong in the automatic modular design of robot swarms (AutoMoDe) [28]. They produce control software for the robots by selecting, fine-tuning, and combining pre-defined software modules into a modular control architecture. These two automatic approaches are characterized by three elements: (i) the robot platform and control architecture on which they operate; (ii) the inverse reinforcement learning algorithm that learns an objective function on the basis of the given demonstrations; and (iii) an optimization process that produces control software by maximizing the learned objective function(s).

A. Robot and control architecture

DTF-MO and DTF-2SO produce control software for the e-puck robot [40] in the form of probabilistic fine-state machines. We consider e-puck robots whose functional capabilities are formally defined by the reference model RM 3 [52]—see Fig. 1. A *reference model* defines the relationship between the control inputs and outputs, and the robot hardware. The e-pucks we consider are endowed with proximity and ground sensors, a range-and-bearing module, a Linux extension board, an omnidirectional camera, two wheels, and RGB LEDs. Due to space limitation, we refer the reader to [53] for details on the robots' hardware. In DTF-MO and DTF-2SO, we adopted the control architecture and pre-defined software modules introduced with AutoMoDe-TuttiFrutti [26]. AutoMoDe-TuttiFrutti generates control software that allows robots to interact with their peers and environment via color signals. We selected this method because it allowed us to conceive composite missions in which a cue—specifically, a color displayed by the walls enclosing the arena in which the robots operate—is available to inform the robots that

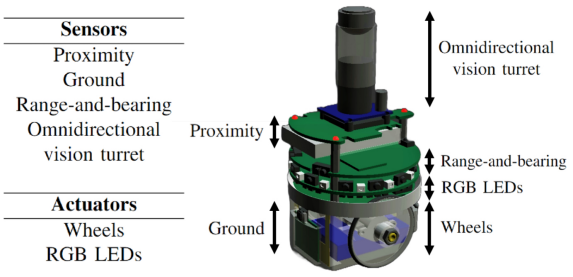

Fig. 1. The e-puck robot and its reference model RM3.

they should transition from one sub-mission to the other. For a detailed description of AutoMoDe-TuttiFrutti, we refer the reader to the original work in which it was introduced [26].

B. Learning the objective function from demonstrations

Like Demo-Cho, DTF-MO and DTF-2SO are based on the algorithm known as *apprenticeship learning via inverse reinforcement learning* [38] where the learning is done in a particular Markov decision process.

A conventional Markov decision process (MDP) is defined by the tuple $(S, A, T, \gamma, D, R^*)$, where S is a finite set of states; A is a set of actions; $T = \{P_{sa}\}$ is a set of state-transition probabilities; $\gamma \in [0, 1)$ is a discount factor; D is the initial-state distribution, from which the state s_0 is drawn; and $R^* : S \rightarrow \mathbb{R}$ is the reward function. In this formalism, it is assumed that the reward function R^* is known and can be used to find an optimal policy. On the other hand, in inverse reinforcement learning, the reward function R^* is unknown and an approximation R should be learned from expert demonstrations. The "true" reward function R^* is assumed to exist and be such that the policy π^* that is optimal with respect to it is the one that would generate the demonstrations provided by the expert.

In apprenticeship learning via inverse reinforcement learning [38] it is assumed that R^* can be expressed as a linear combination of k features $\phi(s)$: $R^*(s) = w^* \cdot \phi(s)$, where $w^* \in \mathbb{R}^k$. The features map the state into a k-dimensional vector: $\phi : S \rightarrow [0, 1]^k$, where S is the set of states and k is the number of features. For every policy π, the expected discounted accumulated feature value vector is $\mu(\pi) = E\left[\sum_{t=0}^{\infty} \gamma^t \phi(s_t) | \pi\right] \in \mathbb{R}^k$. The value of μ for the optimal policy π^*—i.e., the one demonstrated by the expert—is $\mu^* = \mu(\pi^*)$ and is computed using the demonstrations provided. The vector w describing the reward function is iteratively computed via a support vector machine: it is the normal vector of the hyperplane that separates μ^* and the vectors μ obtained in the previous iterations. On the basis of the resulting reward $R_w(s) = w \cdot \phi(s)$, the policy π_w is computed through the optimization process and evaluated. As a result a new value $\mu(\pi_w)$ is obtained and included in the set of previous μ vectors. By iterating the process, the vector w converges to w^*, and the policy π_w to π^*.

In DTF-MO and DTF-2SO, the user is required to provide demonstrations for each sub-mission in the composite mission, which specifically indicate the robots' positions and exemplify the desired spatial distribution of the swarm in the environment. For the apprenticeship learning algorithm, the position of the robots—i.e., swarm state—is mapped to a mission-independent feature vector $\phi(s) \in [0, 1]^k$. This feature vector encapsulates the spatial distribution of the swarm—inter-robot distances—and the location of the swarm in the arena—landmarks-robots distances. It is important to stress here that the feature vector, defined once, depends only on the environment, and not on the mission to perform.

DTF-MO and DTF-2SO apply mapping transformations, which are similar to those of Demo-Cho, to convert the positions of the robots into features describing the spatial relationships between robots and with their environment. In our experimental setup, the twenty e-pucks operate in an arena that comprises three landmarks—see Section IV. The mapping results into 60 features that describe the distance between each robot and the three landmarks, and 20 features that describe the distance between each robot and its closest peer. The features related to landmarks are calculated as follows:

$$\phi_{rl} = \begin{cases} 1, & \text{if robot r is inside landmark l;} \\ 10^{-\frac{2}{d} D_{rl}}, & \text{otherwise.} \end{cases} \quad (1)$$

Here, d is the diameter of the arena and D_{rl} is the distance from robot $r = 1, \ldots, n$ to landmark $l = 1, \ldots, m$, where n and m are the number of robots and the number of landmarks, respectively. For each landmark, $r = 1$ is the closest robot to the landmark itself and $r = 20$ the farthest one.

The features related to distance between robots are calculated as follows:

$$\phi_r = 10^{-\frac{2}{d} D_r}, \quad (2)$$

where D_r is distance between robot $r = 1, \ldots, n$ and its closest peer, and n is the number of robots. Here, $r = 1$ is the robot whose distance to its closest peer is the shortest; and $r = 20$, the one whose distance to its closest peer is the longest. Swarm robots are interchangeable, with mapping based on relative positions. The feature vector, $\phi(s) = (\phi_{11}, \ldots, \phi_{1m}, \ldots, \phi_{n1}, \ldots, \phi_{nm}, \phi_1, \ldots, \phi_n)$, describes the swarm's spatial configuration, regardless of individual identities.

In Demo-Cho, the apprenticeship learning algorithm was limited to learning a single objective function from the demonstrations. In DTF-MO and DTF-2SO, the algorithm learns an objective function for each sub-mission. The composite missions we consider in this study comprise two sub-missions. Therefore, the apprenticeship learning algorithm must learn two objective functions to be optimized, $R_{w_1}(s) = w_1 \cdot \phi(s)$ and $R_{w_2}(s) = w_2 \cdot \phi(s)$, where w_1 and w_2 are the weight vectors that describe the importance given to each feature.

We set a maximum number of iterations as the termination criterion for the apprenticeship learning—the same criterion defined in Demo-Cho. The maximum number of iterations

is chosen to be sufficiently large to allow w_1 and w_2 to converge.

C. Multi-criteria optimization

DTF-MO and DTF-2SO must produce control software for the robots on the basis of the objective functions learned via apprenticeship learning, $R_{w_1}(s)$ and $R_{w_2}(s)$, which define two performance measures to be maximized. However, there is currently no well-defined approach to conducting multi-criteria optimization in the automatic design of robot swarms. We therefore base the multi-criteria design process of DTF-MO and DTF-2SO on two strategies previously adopted in the literature.

DTF-MO conducts a single-objective optimization process that concurrently considers $R_{w_1}(s)$ and $R_{w_2}(s)$. DTF-MO linearly combines the two objectives into a single one: $R(S) = R_{w_1}(s) + R_{w_2}(s) = w_1 \cdot \phi(s) + w_2 \cdot \phi(s)$. This is an approach previously adopted in the swarm robotics literature to aggregate multiple performance measures [23], [26], [54], [55]. Here, the two objective functions are equally weighted as there is no reason to assume *a priori* that one sub-mission is more or less important than the other.

DTF-2SO, on the contrary, conducts single-objective optimization processes that consider $R_{w_1}(s)$ and $R_{w_2}(s)$ separately. This results in two instances of control software, one that maximizes $R_{w_1}(s)$ and another one that maximizes $R_{w_2}(s)$. After the design process ends, we manually assemble the two instances by hard-coding a transition condition from one to the other—according the environmental cue that indicates the transition between sub-missions. This is a similar approach to that adopted in [39].

The design process in DTF-MO and DTF-2SO is based on Iterated F-race [56]—a single-objective optimization algorithm adopted in many existing AutoMoDe methods [22]–[28]. During the design process, DTF-MO and DTF-2SO produce a set of candidate control software instances, each representing a potential solution to the composite mission. In order to select the best performing instance, we assume that the distance in the feature space indicates how close a behavior is to the demonstrations given for each sub-mission. Each instance has two associated distances, one for each objective, $R_{w_1}(s)$ and $R_{w_2}(s)$. DTF-2SO and DTF-2SO compute the L2 norm to aggregate the distance values of an instance into a single measure that can be used to compare instances across the two sub-missions. The smaller the L2 norm, the closer the control software is to the demonstrations. This method has been shown to be effective in feature selection problems [57].

IV. EXPERIMENTAL SETUP

We conducted experiments to assess the performance of DTF-MO and DTF-2SO against the manually designed baseline. In these experiments, the robots operate in an hexagonal arena of $2.60\,\mathrm{m}^2$ with gray floor. The arena comprises three landmarks: a white circular region in the center, and two black triangular regions at the left and right sides. These are the three landmarks considered in the learning process.

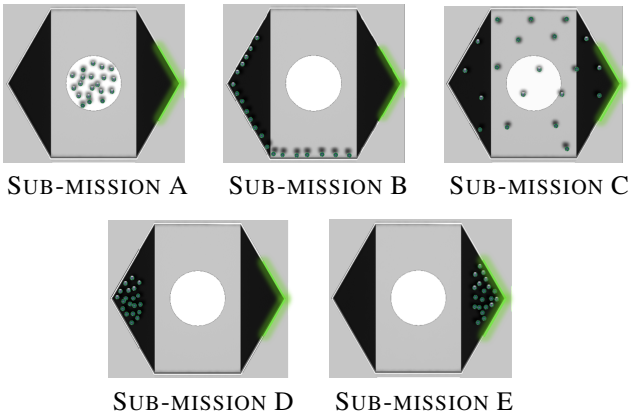

Fig. 2. Experimental arena. The figure indicates the three landmarks, RGB blocks, and a swarm of 20 e-pucks that examplify the demonstrations given for each sub-mission. The green walls do not change color. Other walls switch from red to blue to indicate the transition between sub-missions. This is the cue that the automatic design process must learn to exploit.

The walls of the arena are made of 24 RGB blocks that can change their color at run-time [58]. The 4 blocks at the right corner of the arena constantly display the color green.

The robots have $120\,\mathrm{s}$ to perform the two sub-missions, with $60\,\mathrm{s}$ allocated to each. The arena walls (except for the right corner) are red for the first $60\,\mathrm{s}$ and then turn blue, indicating which sub-mission should be performed. This is the environmental cue we use to manually code the transition between behaviors in the control software produced by DTF-2SO—see Section III. In DTF-MO, the reaction to this cue and the behavior transition must be automatically inferred during the design process. DTF-MO and DTF-2SO extract and evaluate the position of the robots only at the end of each sub-mission—that is, after their allocated $60\,\mathrm{s}$. Fig. 2 shows the experimental arena.

Without a pre-existing theoretical baseline, we developed our own. Utilizing the AutoMoDe-Chocolate modules, we manually designed a control software instance for each sub-mission. Then, we manually assembled the two corresponding instances of a composite mission by using the same manually designed transition as the one used for DTF-2SO.

A. Missions

We conceived five sub-missions that we paired to produce twelve composite missions.[1] Fig. 2 shows the sub-missions as specified by the demonstrations. The sub-missions we conceived are inspired by spatial-organizing behaviors observed in previous automatic design studies [24]–[27], [59]. Although simple, these sub-missions require developing the same basic collective behaviors that would be needed in real-world applications.

For clarity, we also provide a written description of the desired behavior specified by the demonstrations in Fig. 2. SUB-MISSION A (m_{A}): the robots must aggregate in the

[1]By combining the five missions, we could generate twenty sequences. We selected twelve of them for these experiments.

center of the arena, on the white area. SUB-MISSION B (m_{B}): the robots must line in the edges of the arena, except for the two rightmost walls. SUB-MISSION C (m_{C}): the robots must cover all the arena. SUB-MISSION D (m_{D}): the robots must aggregate in the left black area. SUB-MISSION E (m_{E}): the robots must aggregate in the right black area.

The twelve composite missions are organized in pairs, with each pair denoted as $m_{\text{X·Y}}$, where X and Y represent the sub-missions to be executed in sequence. The pairs are: $m_{\text{A·B}}$, $m_{\text{B·A}}$, $m_{\text{C·B}}$, $m_{\text{B·C}}$, $m_{\text{C·D}}$, $m_{\text{D·C}}$, $m_{\text{E·A}}$, $m_{\text{A·E}}$, $m_{\text{B·E}}$, $m_{\text{E·B}}$, $m_{\text{D·E}}$ and $m_{\text{E·D}}$.

B. Protocol

We generated 120 instances of control software with DTF-MO and other 120 with DTF-2SO—10 for each mission. In every design process, we specified the composite mission with 10 demonstrations—five for each associated sub-mission. In all cases, the apprenticeship learning algorithm performed 15 iterations. In every iteration, Iterated F-race had a budget of 100 000 simulations to produce a control software. DTF-MO was given these 100 000 simulations to produce a single instance of control software for the two sub-missions. DTF-2SO was given an budget of 100 000 simulations to produce each of the two instances of control software, required by the two sub-missions. This resulted in an advantage for DTF-2SO, considering that its search space is less than half that of DTF-MO.

We present numerical results with notched box-plots. We use the L2 Norm—see Section III—to quantitatively compare the performance of the instances of control software produced with DTF-MO, DTF-2SO, and the manual baseline. We use heat-map plots to present the distribution of weights, w_1 and w_2, that the apprenticeship learning algorithm assigned to the linear combination of features. In addition, we use scatter plots to study the distribution of the solutions with respect to the distance to each sub-mission and we conduct a visual inspection of the behaviors of the robots to verify their ability to reproduce the given demonstrations.

V. RESULTS AND DISCUSSION

The source code, control software produced, scatter plots, and demonstration videos are available as supplementary material [60].

The numerical results presented in Fig. 3 show that DTF-2SO outperforms DTF-MO and the manual baseline in half of the missions. Specifically, the L2 norm is lower for DTF-2SO in these missions. While the L2 norm gave a general idea, the results of the scatter plots and the visual analysis gave more detailed information. They allowed to see clearly if an approach generated a control software performed well only for one sub-mission or the two of them. Two factors can influence the performance of the methods: (i) the ability to learn the objective functions from the demonstrations, (ii) and the effectiveness of the optimization process in finding suitable control software.

We first investigated the ability of DTF-MO and DTF-2SO to identify the subset of features that is most relevant for each sub-mission, and properly learn a distribution of weights for w_1 and w_2. The heatmap plots showed that, across all missions, the two methods were able to identify the most relevant features during the learning process—see supplementary material [60]. Indeed, DTF-MO and DTF-2SO assigned mission-specific distribution of weights that properly characterized the desired spatial relationship between robots and environment—according to the demonstrations of the two associated sub-missions. We illustrate these results with the weights assigned in mission $m_{D·E}$—see Fig. 4. In SUB-MISSION D, the robots were expected to stay close to the left black landmark and far from the right one. Conversely, in SUB-MISSION E, the robots were expected to remain close to the right black landmark and far from the left one. Fig. 4 shows how both DTF-MO and DTF-2SO put the appropriate emphasis on the related features in each case. These results allowed us to isolate the main factor that caused the performance difference: the optimization process and its ability to produce suitable control software.

Fig. 3 shows that DTF-MO performed better than the manual baseline and equally or better than DTF-2SO in four missions: $m_{C·B}$, $m_{A·B}$, $m_{C·D}$ and $m_{A·E}$. We visually inspected these missions to determine possible reasons for the performance difference with respect to the general results. We observed that DTF-MO was more effective because it designed behaviors that ease the transition from one sub-mission to the other. By concurrently considering the two sub-missions, it not only optimized each part of the composite mission but also the transition phase between them. For example, in SUB-MISSION A of $m_{A·B}$, the robots do not exactly aggregate in the center of the arena—as indicated in the demonstration. Instead, they remain near the borders of the white circle to perceive the walls more easily and react promptly to the cue. Similarly, we observed that in SUB-MISSION D of $m_{C·D}$, the robots not only head towards the left landmark. They also trigger their own cue to signal other robots to follow them—a color-based communication behavior previously observed in AutoMoDe-TuttiFrutti [26].

We also inspected two missions in which neither DTF-MO nor DTF-2SO outperformed the manual baseline. We observed that missions $m_{E·A}$ and $m_{B·A}$ are challenging because of a difficult transition phase between the sub-missions. DTF-MO focuses on the second sub-mission to get the best performance out of it, without risking losing too much time with the transition. In DTF-2SO, the optimization process is conducted independently for each sub-mission and the control software is manually assembled afterward, preventing the generation of behaviors that consider the transition between sub-missions. For example, in $m_{E·A}$, the robots effectively aggregate in the right green corner. However, from this corner, some of them are unable to perceive the cue to start performing mission SUB-MISSION A, and fail to aggregate in the center of the arena—as demonstrated.

Our experiments show that DTF-MO is viable for the automatic design of robot swarms that perform composite missions, which are specified via demonstrations. However,

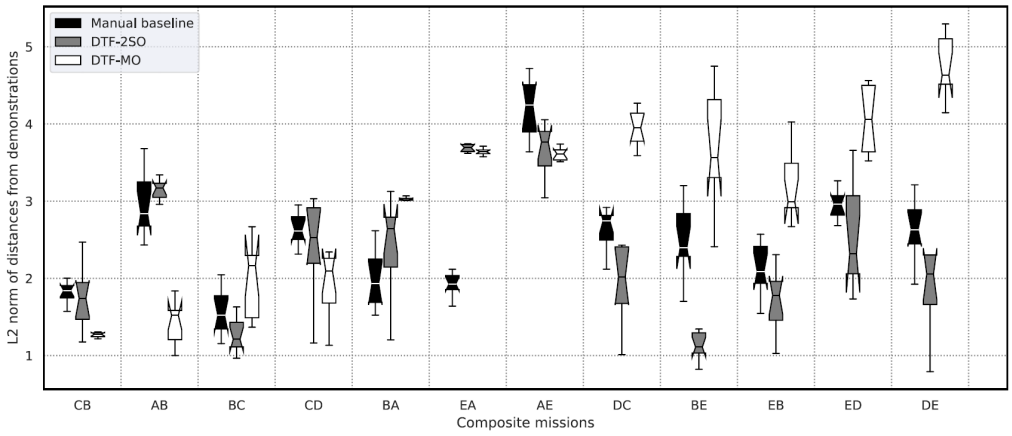

Fig. 3. Experimental results. The lower, the better.

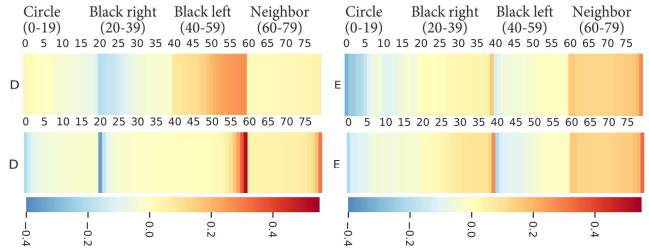

Fig. 4. Heat-map plots for mission $m_{D.E}$ represents the weight associated to each of the 80 features in the objective functions. When the weight is near zero, the feature's impact on the objective function is minimal. Negative weights (blue color) require feature minimization, positive weights (orange color) require maximization. As features are inversely proportional to distances, maximizing a feature minimizes its corresponding distance, and vice versa. DTF-MO (top) and DTF-2SO (bottom) learned for (i) m_{D}, to maximize the distance from the right triangle and minimize it with the left one; (ii) m_{B}, to maximize the distance with the left triangle and the center and minimize it with the right triangle and the peers.

this method is challenged by the increased search space of conducting a design process that tracts the composite mission as a whole. DTF-MO has to design good-performing control software for the two sub-missions, and simultaneously identify the cue and define the transition rule. DTF-2SO, on the contrary, benefits from independently addressing the sub-missions during the design process, and from a higher degree of human intervention: manually assembling the generated control software and hard-coding a transition rule. We argue that this intervention eased the exploration of the search space for DTF-2SO with respect to DTF-MO, and allowed it to perform better. Notwithstanding, DTF-2SO's advantage is challenged when the demonstrations show possibly conflicting positioning for the robots, as it can prevent the swarm from properly reacting to the cue that is hard-coded manually.

In DTF-MO, we used a rather simple multi-criteria optimization strategy and we expect that more advanced alternatives [61] could possibly extend the range of missions it can address. We will devote future work to investigating whether more advanced optimization algorithms available in the literature can bootstrap the design capabilities of DTF-MO, while maintaining the fully automatic nature of its design process. The selection of appropriate optimization algorithms is an open problem in the automatic design of robot swarms [11], [25], [62].

A known limitation of our approach to the design by demonstrations is that it currently only allows for specifying desired static spatial-organizing behaviors—as also noted by Gharbi *et al.* [34]. In the future, we plan to investigate the applicability of our approach in other design problems.

VI. CONCLUSION

In this work, we investigated the problem of automatically designing robot swarms that can perform composite missions. We proposed two approaches based on inverse reinforcement learning and demonstrated their efficacy in generating robot control software. DTF-MO showed the ability to produce control software based only on demonstrations of the desired swarm behavior, whereas DTF-2SO required also human intervention to assemble the control software for each sub-mission. These results contribute to advancing the automatic design of robot swarms towards a more natural way of specifying the missions for the robots. Additionally, our contribution to the literature includes defining a protocol to experiment with the realization of robot swarms that must perform composite missions.

So far, automatic design approaches have focused on addressing missions defined by a single performance measure to be optimized. Here, we show that it is possible to perform missions whose specification defines multiple performance measures. However, increasing the number of performance measures to be considered raises new challenges to the optimization algorithms typically considered in the literature—and therefore, to their ability of generating suitable control software. Our experiments have also identified weaknesses in each approach and suggest avenues for improvement. Particularly, we shed light on different advantages and limitations of using different multi-criteria optimization algorithms in the automatic design of robot swarms.

REFERENCES

[1] 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.

[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] M. Dorigo, M. Birattari, and M. Brambilla, "Swarm robotics," *Scholarpedia*, vol. 9, no. 1, p. 1463, 2014.

[4] M. Rubenstein, A. Cornejo, and R. Nagpal, "Programmable self-assembly in a thousand-robot swarm," *Science*, vol. 345, no. 6198, pp. 795–799, 2014.

[5] J. Werfel, K. Petersen, and R. Nagpal, "Designing collective behavior in a termite-inspired robot construction team," *Science*, vol. 343, no. 6172, pp. 754–758, 2014.

[6] L. Garattoni and M. Birattari, "Autonomous task sequencing in a robot swarm," *Science Robotics*, vol. 3, no. 20, p. eaat0430, 2018.

[7] I. Slavkov, D. Carrillo-Zapata, N. Carranza, X. Diego, F. Jansson, J. Kaandorp, S. Hauert, and J. Sharpe, "Morphogenesis in robot swarms," *Science Robotics*, vol. 3, no. 25, p. eaau9178, 2018.

[8] J. Yu, B. Wang, X. Du, Q. Wang, and L. Zhang, "Ultra-extensible ribbon-like magnetic microswarm," *Nature Communications*, vol. 9, no. 1, p. 3260, 2018.

[9] S. Li, R. Batra, D. Brown, H.-D. Chang, N. Ranganathan, C. Hoberman, D. Rus, and H. Lipson, "Particle robotics based on statistical mechanics of loosely coupled components," *Nature*, vol. 567, no. 7748, pp. 361–365, 2019.

[10] H. Xie, M. Sun, X. Fan, Z. Lin, W. Chen, L. Wang, L. Dong, and Q. He, "Reconfigurable magnetic microrobot swarm: multimode transformation, locomotion, and manipulation," *Science Robotics*, vol. 4, no. 28, p. eaav8006, 2019.

[11] 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.

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

[13] G.-Z. Yang, J. Bellingham, P. E. Dupont, P. Fischer, L. Floridi, R. Full, N. Jacobstein, V. Kumar, M. McNutt, R. Merrifield, B. J. Nelson, B. Scassellati, M. Taddeo, R. Taylor, M. Veloso, Z. L. Wang, and R. J. Wood, "The grand challenges of Science Robotics," *Science Robotics*, vol. 3, no. 14, p. eaar7650, 2018.

[14] 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.

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

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

[17] G. Baldassarre, V. Trianni, M. Bonani, F. Mondada, M. Dorigo, and S. Nolfi, "Self-organized coordinated motion in groups of physically connected robots," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 37, no. 1, pp. 224–239, 2007.

[18] M. Gauci, J. Chen, W. Li, T. J. Dodd, and R. Groß, "Self-organized aggregation without computation," *The International Journal of Robotics Research*, vol. 33, no. 8, pp. 1145–1161, 2014.

[19] 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.

[20] S. Jones, M. Studley, S. Hauert, and A. Winfield, "Evolving behaviour trees for swarm robotics," in *Distributed Autonomous Robotic Systems: The 13th International Symposium*, ser. Springer Proceedings in Advanced Robotics, R. Groß, A. Kolling, S. Berman, E. Frazzoli, A. Martinoli, F. Matsuno, and M. Gauci, Eds., vol. 6. Cham, Switzerland: Springer, 2018, pp. 487–501.

[21] 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.

[22] A. Ligot, K. Hasselmann, and M. Birattari, "AutoMoDe-Arlequin: neural networks as behavioral modules for the automatic design of probabilistic finite state machines," in *Swarm Intelligence: 12th International Conference, ANTS 2020*, ser. Lecture Notes in Computer Science, M. Dorigo, T. Stützle, M. J. Blesa, C. Blum, H. Hamann, M. K. Heinrich, and V. Strobel, Eds., vol. 12421. Cham, Switzerland: Springer, 2020, pp. 109–122.

[23] J. Kuckling, V. van Pelt, and M. Birattari, "AutoMoDe-Cedrata: automatic design of behavior trees for controlling a swarm of robots with communication capabilities," *SN Computer Science*, vol. 3, p. 136, 2022.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

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

[32] D. Bozhinoski and M. Birattari, "Towards an integrated automatic design process for robot swarms," *Open research Europe*, vol. 1, p. 112, 2021.

[33] A. Šošić, W. R. Khuda Bukhsh, A. M. Zoubir, and H. Koeppl, "Inverse reinforcement learning in swarm systems," in *AAMAS '17: Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*. Richland, SC, USA: International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS), 2017, pp. 1413—1421.

[34] 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.

[35] A. Y. Ng, D. Harada, and S. J. Russel, "Policy invariance under reward transformations: theory and application to reward shaping," in *ICML'99: Proceedings of the 16th Annual International Conference on Machine Learning*, I. Bratko and S. Dzeroski, Eds. San Francisco, CA, USA: Morgan Kaufmann Publishers, 1999, pp. 278–287.

[36] P. Abbeel and A. Y. Ng, "Inverse Reinforcement Learning," in *Encyclopedia of Machine Learning*, C. Sammut and G. I. Webb, Eds. Boston, MA: Springer US, 2010, pp. 554–558. [Online]. Available: https://doi.org/10.1007/978-0-387-30164-8417

[37] S. J. Russel, "Learning agents for uncertain environments (extended abstract)," in *COLT' 98: Proceedings of the 11th annual conference on Computational learning theory*, P. Bartlett and Y. Mansour, Eds. New York, NY, USA: ACM, 1998, pp. 101–103.

[38] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proceedings of the twenty-first international conference on Machine learning*, 2004, p. 1.

[39] 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.

[40] 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.

[41] 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.

[42] 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.

[43] K. Hasselmann and M. Birattari, "Modular automatic design of collective behaviors for robots endowed with local communication capabilities," *PeerJ Computer Science*, vol. 6, p. e291, 2020.

[44] 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.

[45] J. Kuckling, "Recent trends in robot learning and evolution for swarm robotics," *Frontiers in Robotics and AI*, vol. 10, p. 1134841, 2023.

[46] M. J. Krieger, J.-B. Billeter, and L. Keller, "Ant-like task allocation and recruitment in cooperative robots," *Nature*, vol. 406, no. 6799, pp. 992–995, 2000.

[47] T. Schmickl, R. Thenius, C. Moslinger, J. Timmis, A. Tyrrell, M. Read, J. Hilder, J. Halloy, A. Campo, C. Stefanini, *et al.*, "Cocoro–the self-aware underwater swarm," in *2011 Fifth IEEE Conference on Self-Adaptive and Self-Organizing Systems Workshops*. IEEE, 2011, pp. 120–126.

[48] E. C. Ferrer, T. Hardjono, A. Pentland, and M. Dorigo, "Secure and secret cooperation in robot swarms," *Science Robotics*, vol. 6, no. 56, p. eabf1538, 2021.

[49] M. J. B. Krieger, J.-B. Billeter, and L. Keller, "Ant-like task allocation and recruitment in cooperative robots," *Nature*, vol. 406, pp. 992–995, 2000.

[50] S. Nouyan, R. Groß, M. Bonani, F. Mondada, and M. Dorigo, "Teamwork in self-organized robot colonies," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 4, pp. 695–711, 2009.

[51] P. C. Fishburn, *Utility Theory for Decision Making*, ser. Publications in Operations Research. New York, NY, USA: John Wiley & Sons, 1970.

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

[53] 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.

[54] 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.

[55] 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.

[56] 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.

[57] R. Salman, A. Alzaatreh, and H. Sulieman, "The stability of different aggregation techniques in ensemble feature selection," *Journal of Big Data*, vol. 9, no. 1, pp. 1–23, 2022.

[58] D. Garzón Ramos, M. Salman, K. Ubeda Arriaza, K. Hasselmann, and M. Birattari, "MoCA: a modular RGB color arena for swarm robotics experiments," IRIDIA, Université libre de Bruxelles, Brussels, Belgium, Tech. Rep. TR/IRIDIA/2022-014, 2022.

[59] G. Spaey, M. Kegeleirs, D. Garzón Ramos, and M. Birattari, "Evaluation of alternative exploration schemes in the automatic modular design of robot swarms," in *Artificial Intelligence and Machine Learning: BNAIC 2019, BENELEARN 2019*, ser. Communications in Computer and Information Science, B. Bogaerts, G. Bontempi, P. Geurts, N. Harley, B. Lebichot, T. Lenaerts, and G. Louppe, Eds. Cham, Switzerland: Springer, 2020, vol. 1196, pp. 18–33.

[60] J. Szpirer, D. Garzón Ramos, and M. Birattari, "Automatic design of robot swarms that perform composite missions: an approach based on inverse reinforcement learning," https://iridia.ulb.ac.be/supp/IridiaSupp2023-003, 2024.

[61] M. T. M. Emmerich and A. H. Deutz, "A tutorial on multiobjective optimization: fundamentals and evolutionary methods," *Natural Computing*, vol. 17, no. 3, pp. 585–609, 2018.

[62] J. Kuckling, T. Stützle, and M. Birattari, "Iterative improvement in the automatic modular design of robot swarms," *PeerJ Computer Science*, vol. 6, p. e322, 2020.