frontiers | Frontiers in Robotics and AI

TYPE Review
PUBLISHED 24 April 2023
DOI 10.3389/frobt.2023.1134841

OPEN ACCESS

EDITED BY
Phil Husbands,
University of Sussex, United Kingdom

REVIEWED BY
Larry Bull,
University of the West of England, United Kingdom
Andrea Roli,
University of Bologna, Italy

*CORRESPONDENCE
Jonas Kuckling,
jonas.kuckling@ulb.be

SPECIALTY SECTION
This article was submitted to Robot Learning and Evolution, a section of the journal Frontiers in Robotics and AI

RECEIVED 30 December 2022
ACCEPTED 21 March 2023
PUBLISHED 24 April 2023

CITATION
Kuckling J (2023), Recent trends in robot learning and evolution for swarm robotics.
Front. Robot. AI 10:1134841.
doi: 10.3389/frobt.2023.1134841

Recent trends in robot learning and evolution for swarm robotics

Jonas Kuckling*

IRIDIA, Université Libre de Bruxelles, Brussels, Belgium

Swarm robotics is a promising approach to control large groups of robots. However, designing the individual behavior of the robots so that a desired collective behavior emerges is still a major challenge. In recent years, many advances in the automatic design of control software for robot swarms have been made, thus making automatic design a promising tool to address this challenge. In this article, I highlight and discuss recent advances and trends in offline robot evolution, embodied evolution, and offline robot learning for swarm robotics. For each approach, I describe recent design methods of interest, and commonly encountered challenges. In addition to the review, I provide a perspective on recent trends and discuss how they might influence future research to help address the remaining challenges of designing robot swarms.

KEYWORDS

swarm robotics, robot evolution, robot learning, automatic design, neuro-evolution, automatic modular design, embodied evolution, imitation learning

1 Introduction

Robot swarms are decentralized systems of relatively simple robots that only rely on local information to operate (Beni, 2005; Şahin, 2005; Brambilla et al., 2013; Dorigo et al., 2014; Hamann, 2018). Like animal swarms in nature, a robot swarm is a group of robots that are efficient at performing tasks due to their cooperation. Robot swarms are multi-robot systems that exhibit some particular characteristics. They are decentralized and highly redundant. The high redundancy requires that there is no role in the swarm that can only be executed by a single robot[1]. Furthermore, in a robot swarm, there exists no single central point of control (neither internal nor external to the swarm), as a centralized point of control would be a single point of failure. Therefore, complex collective behaviors, such as task allocation, cannot be planned and orchestrated by an operator. Instead, the swarm is required to be self-organizing: the collective behavior of the swarm must emerge from the interactions between the individual robots. Additionally, the robots in the swarm are relatively simple (both in terms of hardware and software) with respect to the task they perform and have only local sensing and communication capabilities.

1 Classically, a robot swarm is a homogeneous system—i.e., all robots have the same capabilities and execute the same software. There have been examples of heterogeneous robot swarms (Dorigo et al., 2013), in which parts of the swarm are specialized in such a way that their role cannot be performed by some of the other robots in the swarm. Yet, in these examples, heterogeneous swarms are also redundant to some degree, as each role has at least several robots being able to perform it.

These inherent characteristics of robot swarms promote the development and implementation of robotic systems that exhibit desirable properties (Şahin, 2005; Dorigo et al., 2014; Hamann, 2018). The self-organized nature of robot swarms promotes the design of flexible systems: the swarm can adapt to different and potentially changing environments. Additionally, the redundancy of the swarm facilitates the creation of systems that are fault tolerant. The failure of any individual robot (or sometimes significant portions of the swarm) will not prevent the swarm from achieving its task. Lastly, as robots only interact with their immediate neighboring peers, swarms are scalable systems. That is, the addition or removal of robots from the swarm does not significantly affect the performance of the swarm. Thanks to these properties, swarm robotics is considered a prominent approach to control large groups of autonomous robots (Rubenstein et al., 2014; Werfel et al., 2014; Mathews et al., 2017; Garattoni and Birattari, 2018; Slavkov et al., 2018; Yu et al., 2018; Li et al., 2019; Xie et al., 2019; Dorigo et al., 2020) and has been recently highlighted as one of the grand challenges of robotics research for the upcoming years (Yang et al., 2018). The use of robot swarms has been proposed for coordinating groups of robots in missions in dynamic or unknown environments, such as for space exploration, search and retrieval in disaster situations, or agricultural applications (Carrillo-Zapata et al., 2020; Dorigo et al., 2020). While no real-world application of swarm robotics exists yet, they are projected to be developed within the next ten to 15 years (Dorigo et al., 2020).

Although the realization of robot swarms offers several advantages, their decentralized and self-organized nature makes them challenging to design. The requirements for the desired behavior of the swarm are usually expressed at the collective level, but it is not possible to program the swarm directly. Instead, the individual robots need to be programmed in such a way that the desired collective behavior arises. The problem is that each robot can only act based on the local information that it can perceive. When programming the robots, the designer needs to predict how the local behaviors and local interactions between robots will contribute to the emergence of the desired collective behavior.

Swarm robotics originated from the application of bio-inspired swarm intelligence principles to robotics (Beckers et al., 1994; Beni, 2005). Since then, swarm robotics has moved towards a more matured engineering discipline—often referred to as swarm engineering (Winfield et al., 2005; Brambilla et al., 2013). Swarm engineering concerns the creation of arbitrary (not necessarily bio-inspired) collective behaviors for a robot swarm. The most common approach to the design of robot swarms is manual design: a human designer manually implements the control software for the robots. The designer can refine the control software through a trial-and-error process until they find the result satisfactory. While this design process often yields reasonably good results, it can be error-prone, costly, time-consuming and the quality of the control software strongly depends on the expertise of the designer. Furthermore, there is no guarantee that the performance will reach a satisfactory level within any reasonably available time budget. Several principled methods and design patterns for designing collective behaviors have been developed to overcome the limitations of pure trial-and-error design (Halloy et al., 2007; Soysal and Şahin, 2007; Hamann and Wörn, 2008; Yamins and Nagpal, 2008; Berman et al., 2009; Kazadi, 2009; Prorok et al., 2009; Berman et al., 2011; Beal et al., 2012; Brambilla et al., 2014; Reina et al., 2015; Lopes et al., 2016; Pinciroli and Beltrame, 2016; Hamann, 2018). Yet, these methods are restricted to specific assumptions and no generally applicable methodology has been proposed yet (Brambilla et al., 2013; Francesca and Birattari, 2016; Schranz et al., 2021).

An alternative to principled methods are (semi-)automatic design methods, which are built upon techniques such as robot evolution or robot learning. In semi-automatic design, a human designer remains in the loop during the development and optimization of the control software. That is, the human designer can observe and intervene in the design process, if necessary. For example, the human designer could observe the result found by the design process, change some parameters used in the algorithms to produce the control software and restart them with the new parameter values. The semi-automatic design terminates when the human designer is satisfied with the generated control software. In a research setting, semi-automatic design allows to assess the underlying feasibility of these design methods. However, in practice, semi-automatic design exhibits similar drawbacks as manual design. Namely, the quality of the generated control software depends on the human designer and their ability to steer the design process. In fully automatic design (Birattari et al., 2019), no human intervention beyond the mission specification is possible (Birattari et al., 2020). That is, the design process runs completely automatic until it terminates with the generation of an appropriate instance of control software. (Semi-)Automatic design methods can be further categorized into online and offline methods (Bredeche et al., 2018; Birattari et al., 2020). In online design, the design process is executed while the swarm performs its mission in the target environment, whereas in offline design, the design process is executed before the swarm is deployed to perform its mission.

In this work, I discuss recent advances in robot evolution and robot learning in the context of swarm robotics (see **Tables 1**–**7** for an index of the considered design methods). The work is organized as follows. In **Section 2**, I present offline design methods that rely on evolutionary algorithms or related techniques. In **Section 3**, I present online design methods. In **Section 4**, I present offline design methods based on robot learning. In **Section 5**, I provide a perspective on important open questions on the application of robot learning and evolution in swarm robotics.

2 Robot evolution

The application of evolutionary robotics principles (Husbands and Harvey, 1992; Nolfi and Floreano, 2000) to swarm robotics is called *evolutionary swarm robotics* (Trianni, 2008; Nolfi, 2021). In evolutionary swarm robotics, the control software of the robots is generated through an artificial evolutionary process. Unless otherwise specified, the same generated control software is uploaded to each robot to be executed individually. The evolutionary process optimizes instances of control software with respect to a mission-specific *objective function*, often also called fitness function. The objective function is used to assess the quality of instances of control software, and in a way, provides selection pressure to direct the optimization process. Poorly performing instances are discarded and the well performing ones are *selected* to generate new instances

TABLE 1 Overview of selected neuro-evolutionary research in swarm robotics.

Publication	Swarm composition	Mission	Network topology	Algorithm	Sim.	Real.
Trianni and Nolfi (2011)	3 s-bot robots	Synchronization	Single layer perceptron	Evolutionary algorithm	•	
Gauci et al. (2014a)	10 e-puck robots	Aggregation	Fully-recurrent network	Classical Evolutionary Programming	•	•
Duarte et al. (2016)	5–10 aquatic drones	Homing, dispersion, clustering, area monitoring	Feedforward network	NEAT	•	•
Gomes et al. (2019)	1 aerial, 1 ground robot	Foraging	Feedforward network	NEAT	•	
Hasselmann et al. (2021)	20 e-puck robots	Aggregation, homing foraging sheltering gate passing	Single layer perceptron, multi-layer perceptron	CMA-ES, xNES, NEAT, evolutionary algorithm	•	•
van Diggelen et al. (2022)	14 ground robots	Gradient following	Fully connected reservoir network	Differential evolution	•	

TABLE 2 Overview of selected automatic modular design research in swarm robotics.

Publication	Swarm composition	Mission	Architecture	Modules	Sim.	Real.
Hecker et al. (2012)	1–3 ground robots	Foraging	Finite-state machine	Manually implemented	•	•
Duarte et al. (2014)	50 aquatic drones	Patrolling	Finite-state machine	Evolved continuous-time recurrent network	•	
Francesca et al. (2014)	20 e-puck robots	Aggregation, foraging	Finite-state machine	Manually implemented	•	•
Ferrante et al. (2015)	4 foot-bot robots	Foraging	Rule set	Manually implemented	•	
Jones et al. (2018)	25 kilobot robots	Foraging	Behavior tree	Manually implemented	•	•
Neupane and Goodrich (2019)		Foraging, co-op. transport, nest maintenance	Behavior tree	Manually implemented	•	
Ligot et al. (2020a)	20 e-puck robots	Aggregation, foraging	Finite-state machine	Evolved feedforward networks	•	•

TABLE 3 Overview of selected novelty search research in swarm robotics.

Publication	Swarm composition	Mission	Sim.	Real.
Gomes et al. (2013)	5–7 e-puck robots	Aggregation, resource sharing	•	
Gomes et al. (2017)	3–8 ground robots	Predator-prey, herding, cooperative foraging	•	
Gomes and Christensen (2018)	5–10 ground robots	Aggregation, clustering, coverage, border coverage, dispersion, phototaxis, flocking	•	
Hasselmann et al. (2023)	20 e-puck robots	Foraging, aggregation, sheltering	•	•

TABLE 4 Overview of selected other evolution-based research in swarm robotics.

Publication	Swarm composition	Mission	Approach	Sim.	Real.
Hamann (2014)	20 particles	Collective motion	Minimizing surprise	•	
Gauci et al. (2014b)	5–50 e-puck robots	Clustering	Computation-free control	•	•
Trianni and López-Ibáñez (2015)	6–10 foot-bot robots	Flocking, collaboration	Multi-objective optimization	•	
Kaiser and Hamann (2019)	100 grid-world agents	Collective motion	Minimizing surprise	•	

TABLE 5 Overview of selected embodied evolution research in swarm robotics.

Publication	Swarm composition	Mission	Algorithm	Controller update	Sim.	Real.
Bianco and Nolfi (2004)	64 s-bot robots	Self-assembly	Not identified	Encounter, time-out	•	
Prieto et al. (2010)	8 e-puck robots	Cleaning	r-ASiCo	Energy		•
Bredeche et al. (2012)	9–100 e-puck robots	Foraging	mEDEA	Energy, time-out	•	•
Silva et al. (2015)	5 e-puck robots	Aggregation, phototaxis, collective motion	odNEAT	Energy	•	
Jones et al. (2019)	9 e-puck robots	Collective transport	Parallel island model distributed evolution	Time-out		•
Cambier et al. (2021)	25–200 kilobot robots	Aggregation	Cultural evolution	Encounter	•	•

TABLE 6 Overview of selected multi-agent reinforcement learning research in swarm robotics.

Publication	Swarm composition	Mission	Algorithm	State-space	Sim.	Real.
Matarić (1997)	4 IS Robotics R2 robots	Foraging	Q-learning	Individual		•
Hüttenrauch et al. (2019)	2–10 ground robots	Rendez-vous, predator-prey	Trust Region Policy Optimization	Individual	•	
Bloom et al. (2022)	4–8 foot-bot robots	Collective transport	ADAM	Individual	•	

through *recombination* and *mutation*. The methods presented in this section are automatic offline design methods. That is, methods in which the design process is executed in a centralized manner using simulations and before the robots are deployed. For design methods that run the evolutionary process directly on the robots, see **Section 3**.

In the context of swarm robotics, robot evolution is the most studied automatic design approach. Indeed, evolutionary swarm robotics has been used to create control software for robot swarms in a wide variety of mission such as foraging, collective transport, or pattern formation (Brambilla et al., 2013; Schranz et al., 2020). Traditionally, evolutionary swarm robotics has relied on *neuro-evolution*—the control software in the form of an artificial neural network is optimized using a centralized evolutionary algorithm (see Section 2.1). Other related approaches include *automatic modular design* (see Section 2.2) and *novelty-search-based design* (see Section 2.3). In automatic modular design, the control software is composed of modules that are assembled into more complex control architectures, such as finite-state machines or behavior trees. In novelty-search-based design methods, the selection pressure does not arise from the mission-specific objective function, but rather from a metric of behavioral novelty. While evolutionary swarm robotics design methods have demonstrated promising results in the past, they still face some important challenges that remain unsolved: notably, the generation of control software that is robust to the reality gap and the engineering of appropriate objective functions that can produce a desired collective behavior.

Trianni et al. (2014) and Francesca and Birattari (2016) provide overviews of robot evolution in the context of swarm robotics. For reviews of evolutionary robotics in the single-robot case, see Doncieux et al. (2011), Bongard (2013), Trianni (2014), Doncieux et al. (2015), and Silva et al. (2016).

TABLE 7 Overview of selected imitation learning research in swarm robotics.

Publication	Swarm composition	Mission	Algorithm	Demonstration	Sim.	Real.
Li et al. (2016)	5–11 e-puck robots	Aggregation, object clustering	Turing learning	Motion trajectories	•	•
Šošić et al. (2017)	200 particles	Synchronization	Inverse reinforcement learning	Motion trajectories	•	
Alharthi et al. (2022)	20 ground robots	Not identified	Behavior cloning	Video recordings	•	
Gharbi et al. (2023)	20 e-puck robots	Aggregation, dispersion, sheltering	Apprenticeship learning	Robot positions	•	•

2.1 Neuro-evolution

Neuro-evolution: Robots are controlled by an artificial neural network that maps sensor inputs to actuator outputs. The weights of the neural network, and possibly its topology, are optimized using an evolutionary algorithm with regard to a mission-specific objective function. The design process results in a single well-performing instance of control software.

Neuro-evolution is one of the earliest automatic design methods in swarm robotics (Dorigo et al., 2003; Quinn et al., 2003; Trianni et al., 2003). In this approach, neural networks are used as *black-box* controllers, and the search performed by the evolutionary algorithm does not require domain-specific heuristic information. For this reason, neuro-evolutionary design methods are expected to allow the design of control software with no domain knowledge. For a review of early neuro-evolutionary design methods, see Brambilla et al. (2013).

More recently, several authors have focused on systematically using neuro-evolution to design control software for various robotic platforms—mainly targeting those that could be possibly used in real-world deployments. For example, Trianni and Nolfi evolved a perceptron network to synchronize the movement of a swarm of s-bot robots (Trianni and Nolfi, 2011). Duarte et al. (2016) used NEAT (Stanley and Miikkulainen, 2002) to design control software a swarm of aquatic robots performing tasks such as homing or dispersion (Duarte et al., 2016). Gomes et al. (2019) generated control software for robot teams composed of aerial and ground robots in a foraging task. Hasselmann et al. (2021) compared NEAT, xNES (Glasmachers et al., 2010) and CMA-ES (Hansen and Ostermeier, 2001) to generate control software for a swarm of e-puck (Mondada et al., 2009) robots in five different missions such as aggregation, homing, shelter, foraging, and gate passing (Hasselmann et al., 2021). In a different research direction, researchers have investigated the minimal requirements to evolve specific collective behaviors. For example, Gauci et al. (2014a) evolved a recurrent neural network to perform aggregation. In their study, the authors tested their control software on robots with minimal capabilities: each robot had a single binary sensor that controlled the speed of its two wheels. van Diggelen et al. (2022) evolved a gradient following behavior. Notably, the robots could perceive only the local value of the gradient, not its direction, and they could not communicate with other robots.

Neuro-evolutionary approaches have shown many promising results. Yet, two main challenges remain in the field: fitness engineering and the reality gap. The first challenge is fitness engineering, or how to produce appropriate objective functions to drive the evolutionary process. It is well understood that incorrectly defined objective functions pose two challenges to the evolutionary process: *bootstrapping* and *deception* (Silva et al., 2016). The issue of bootstrapping arises when the objective function fails to apply meaningful selection pressure in low-performance regions of the search space. As a result, the design process explores the low-performance regions in an undirected manner and is unable to converge towards higher performance regions of the search space. The issue of deception describes the case in which the objective function contains easily reachable local optima. In this case, the design process can easily converge towards the local optima and will result in the generation of a suboptimal collective behavior. These two issues can usually be overcome by introducing *a priori* knowledge into the objective function (fitness engineering) (Trianni and Nolfi, 2011; Divband Soorati and Hamann, 2015; Silva et al., 2016). However, the necessity of *a priori* knowledge conditions the effectiveness of a neuro-evolutionary design method; as it will largely depend on the expertise of the designer of the objective function. The second challenge of neuro-evolution is the reality gap. The reality gap are the inescapable differences between the design and deployment environment, and often manifests in a performance drop when designing control software in simulation and assessing it on real robots. Yet, not all design methods are affected similarly by the reality gap, and it is therefore imperative to assess all automatic design methods not only in simulation but on real robots (Birattari et al., 2019). In the context of neuro-evolution, Hasselmann et al. investigated the effects of the reality gap on different neuro-evolutionary design methods (Hasselmann et al., 2021). They showed that, without further mitigation strategies or mission-specific adaptations, sophisticated neuro-evolutionary design methods perform similarly poor in reality as a simple perceptron network.

2.2 Automatic modular design

Automatic modular design: Robots are controlled by an instance of control software assembled from modules. Typical architectures of the control software include finite-state machines and behavior trees. An optimization algorithm possibly assembles the modules within the architecture and further fine-tunes the parameters of the modules, according to a mission-specific objective function. The design process results in a single well-performing instance of control software.

Neuro-evolution enables, in practice, the design of control software without prior domain knowledge. Yet, in cases that domain knowledge is available, it might be incorporated into the design method to achieve better results. Instead of relying on artificial neural networks, automatic modular design methods generate control software that is composed of software modules that are assembled into a more complex control architecture—e.g., finite-state machines or behavior trees (Colledanchise and Ögren, 2018). Through the choice and implementation of these modules, domain knowledge can be incorporated into the design process.

Duarte et al. (2014) manually decomposed a complex object removal task into simpler subtasks. They evolved continuous-time recurrent neural networks that were then assembled, in a modular way, into a hierarchical controller (according to the manual decomposition). Ferrante et al. (2015) used grammatical evolution to design control software for a foraging scenario with task allocation. They designed behavioral rules from basic behavioral and conditional modules. Hecker et al. (2012) used a genetic algorithm to optimize a finite-state machine that controls the behavior of robots in a foraging swarm. The authors pre-programmed an initial finite-state machine, which was inspired by the foraging behavior observed in ants. They used the genetic algorithm to optimize parameters of the finite-state machine that were not chosen at design time. Francesca et al. (2014) proposed AutoMoDe-Vanilla, an automatic modular design method that assembles finite-state machines out of a set of twelve handcrafted modules. Several flavors (i.e., implementations) of AutoMoDe have been proposed to study different elements of the design process, such as different module sets (Ligot et al., 2020a; Garzón Ramos and Birattari, 2020; Hasselmann and Birattari, 2020; Spaey et al., 2020; Mendiburu et al., 2022), hardware-software co-design (Salman et al., 2019), or optimization algorithms (Kuckling et al., 2020a; Kuckling et al., 2020b; Cambier and Ferrante, 2022). Besides finite-state machines, behavior trees have recently gained attention in the literature on automatic modular design. They offer several advantages over finite-state machines, like enhanced modularity and better human readability (Colledanchise and Ögren, 2018). Jones et al. (2018) evolved behavior trees for a foraging swarm of kilobot (Rubenstein et al., 2014) robots. Kuckling et al. investigated the use of behavior trees within the AutoMoDe framework (Kuckling et al., 2018; Kuckling et al., 2020a; Ligot et al., 2020b; Kuckling et al., 2022). Neupane and Goodrich used grammatical evolution to design software for a swarm of 100 robots performing a foraging task (Neupane and Goodrich, 2019).

Automatic modular design methods are an emerging field of research with promising prospects. Preliminary results indicate that they are a viable alternative to neuro-evolutionary design methods, with comparable performance and better transferability between simulation and real robots. However, this advantage comes at the cost of devoting effort to specify the modules. An artificial neural network can map all possible sensory inputs to all possible actuator outputs. As a result, neuro-evolutionary design methods can be used to design control software to perform any mission that is within the capabilities of the robots. In the case of automatic modular design, a human designer must manually implement the modules. The choice of modules implicitly restricts the space of possible missions that can be addressed by an automatic modular design method (Garzón Ramos and Birattari, 2020). If the set of modules is too limited, the design method would only produce satisfactory results for the mission it was conceived for, and the design space might not contain well-performing instances of control software for other missions. In other words, the design method will underperform in most cases. In this situation, the underperforming method can be accepted as it is or it will become necessary to develop a new design method—which ultimately turns into a manual design method rather than an automatic one. An important question to be addressed is, therefore, how to develop general automatic modular design methods that still remain robust to the reality gap.

2.3 Novelty search and quality diversity algorithms

Novelty search: Robots are controlled by an instance of control software in an arbitrary form, although commonly an artificial neural network is used. Instead of optimizing a mission-specific objective function, novelty search algorithms are selecting for control software that exhibits behavioral novelty with regard to previously encountered behaviors. The design process results in a set of behaviorally diverse instances of control software.

Quality diversity algorithms: Robots are controlled by an instance of control software in an arbitrary form. The design process considers two criteria: the quality with respect to a mission-specific objective function and the behavioral novelty with respect to previously encountered instances of control software. The design process returns either a single well-performing instance of control software or a set of diverse and relatively well-performing instances of control software.

Some recent studies focus on the application of novelty search in swarm robotics. Instead of optimizing a mission-specific performance measure, novelty search generates a set of behaviorally diverse instances of control software (Lehman and Stanley, 2011). This approach promises to avoid the issue of deception in objective function engineering. The design method avoids premature convergence by optimizing behavioral diversity instead of the mission-specific objective function.

Gomes et al. (2013) used novelty search to generate aggregation and resource sharing behaviors in a swarm. Additionally, the authors combined the novelty metric with a performance metric to overcome limitations where novelty search could not escape

large, low-performance regions of the search space. In a follow-up work, Gomes et al. (2017) applied novelty search to co-evolutionary problems. Gomes and Christensen also investigated how to generate task-agnostic behavior repertoires using novelty search (Gomes and Christensen, 2018). Hasselmann et al. (2023) proposed AutoMoDe-Nata, an automatic modular design method that uses novelty search to create basic behavioral modules, which then are combined into probabilistic finite-state machines.

In the papers mentioned above, novelty search-based methods have shown to generate simple swarm robotics behaviors. Yet, for more complex collective behaviors, novelty search has not produced control software that performs as well as control software generated with a mission-specific objective function (Gomes and Christensen, 2018; Hasselmann et al., 2023). The outcome of novelty search methods is not a single instance of control software but a set of them—each of which is a behavior with sufficiently different traits. Therefore, the design method must also include a strategy to select the most appropriate behavior of the set (either manually or automatically). Quality diversity algorithms (Pugh et al., 2016) combine the benefits of novelty search with the directed search of evolutionary robotics. Beyond the difficulties of generating complex collective behaviors, novelty search further faces the challenge of defining characteristic traits that describe the collective behavior of the robots. So far, the selection of behavioral characteristics has been done *ad hoc* (Gomes et al., 2013; Gomes and Christensen, 2018; Hasselmann et al., 2023). However, this *ad hoc* selection requires expertise and it can result in potential drawbacks to the design process when done incorrectly. Vast behavior spaces defined by too general behavior characteristics will contain dimensions unrelated to the mission at hand, and they will cause the novelty search to perform poorly (Gomes et al., 2014). Furthermore, it remains open whether the behavioral characteristics of a collective behavior should be defined on a collective level, on an individual level, or with a combination of the two.

2.4 Other approaches

As discussed previously, a major issue in evolutionary swarm robotics is the definition of an appropriate objective function. Other researchers have addressed this issue besides those that focus on novelty search—although they are less prominent in the literature.

Multi-objective optimization: Robots are controlled by an instance of control software in an arbitrary form. Instead of optimizing a single mission-specific objective function, several objectives are considered at the same time. The design process results in a set of non-dominated instances of control software.

Trianni and López-Ibáñez (2015) investigated the use of an evolutionary multi-objective optimization algorithm in a strictly collaborative mission. Next to the (singular) objective of the mission, the authors specified a secondary auxiliary objective to overcome the convergence to certain sub-optimal behaviors—although the auxiliary conflicted with the main objective. They showed that multi-objective optimization indeed avoided premature convergence, and that properly chosen auxiliary objectives have the potential to overcome the bootstrap problem.

Minimizing surprise: Robots are controlled by two artificial neural networks. The first neural network is an action network that maps sensor inputs to actuator outputs; the other is a predictor network that maps sensor inputs to the predicted sensor inputs of the next control step. An evolutionary algorithm is used to optimize both neural networks together, minimizing only the prediction error of the predictor network. The design process results in a single instance of control software that minimizes the prediction error of the predictor network.

Kaiser and Hamann investigated an approach named "minimizing surprise." Inspired by the *free energy principle* (Friston, 2010), Hamann used offline evolution to generate control software in the form of two neural networks, a prediction network and an action network (Hamann, 2014). The action network controlled the robot, whereas the prediction network predicted the next sensor state. The design process aimed to minimize the prediction error. Results showed that, despite not selecting for swarm behaviors, basic self-organizing collective behaviors emerged during the design process. Kaiser and Hamann extended their work and proposed a system to systematically engineer self-organizing assembly behaviors using the "minimizing surprise" approach (Kaiser and Hamann, 2019).

Computation-free control: Robots are controlled by a look-up table that contains the actuator output for every possible sensor input. The look-up table is optimized with respect to a mission-specific objective function. Typically, CMA-ES is used as optimization algorithm, but other algorithms are possible (e.g., exhaustive search). The design process results in a single well-performing instance of control software.

Gauci et al. (2014b) studied the emergence of collective behaviors for robots with minimal capabilities. In their study, the robots only had a single line-of-sight sensor and could set their velocity based on the discrete readings of this sensor. The authors used CMA-ES to optimize the mappings of the sensor to velocities in missions such as clustering, shepherding (Özdemir et al., 2017; Dosieah et al., 2022), decision making (Özdemir et al., 2018), and coverage (Özdemir et al., 2019).

The approaches described in this section are promising alternatives for the design of control software for robot swarms, but they require further research before they will become robust engineering techniques. Multi-objective design methods might overcome issues of deception and bootstrapping, similar to novelty search. However, they do not require the definition of behavioral characteristics. Instead, a secondary mission-specific objective is defined. This also poses a major challenge, as the definition of secondary objectives requires knowledge of any undesired behaviors. Minimizing surprise has shown to generate spatially organized behaviors in which the sensors states are stable. By defining partial expected sensor readings, generated instances of control software can be biased towards desired behaviors, yet other less desired behaviors were still generated. More research will be necessary to develop techniques to reliably generate desired behaviors (or classes of desired behaviors)

via minimizing surprise. Similarly, computation-free control has been successful in some relatively simple missions, but further research will be necessary to show its viability in more complex missions.

3 Embodied evolution and social learning

> *Embodied evolution*: Robots are controlled by instance of control software in an arbitrary form, though, typically, an artificial neural network is chosen. While performing the mission, the robots also run a decentralized evolutionary algorithm and periodically select a new instance of control software to execute. Repeatedly during the execution, the robots may exchange information about the performance of their executed instances of control software. The design process results in a single well-performing instance of control software.

Online design processes in which each robot in the swarm executes part of a distributed evolutionary algorithm are called embodied evolution. For example, in the design process, every robot in the swarm is initialized with and maintains its own set of genomes. From this genome set, each robot selects one genome and executes the control software encoded in it. Periodically, all robots exchange genomes among each other (which may be subjected to mutation or crossover operations) and they select a new genome to execute. Embodied evolution provides important advantages with respect to offline approaches (Watson et al., 2002). As evolution takes place directly in the mission environment, transferability is no concern. Besides, the distributed nature of the design process allows exploring different solutions in parallel. The parallelization of the design process allows to speed up the production of control software if compared with centralized online evolutionary methods.

Recent works (Heinerman et al., 2015; Silva et al., 2017; Bredeche and Fontbonne, 2021) have framed the concepts behind embodied evolution as a form of robot learning. The authors argue that embodied evolution is conceptually closer to robot learning (see **Section 4**), as the robots are updating their control software while performing the mission. Yet, the most common technique to implement embodied evolution remains the application of evolutionary algorithms.

Only a few studies have been conducted using embodied evolution in swarm robotics. For example, Bianco and Nolfi (2004) investigated an embodied evolutionary approach in which robots share their genomes when physically connecting to other robots. Prieto et al. (2010) used embodied evolution to program a swarm of e-puck robots in a cleaning task. Bredeche et al. (2012) investigated the adaptivity of open-ended evolution to changes in the environment. Silva et al. (2015) developed an online, distributed version of NEAT (Stanley and Miikkulainen, 2002) and used it to evolve control software in three missions. Jones et al. (2019) evolved behavior trees for a collective pushing task. Cambier et al. (2021) used an evolutionary language model to tune the parameters of a probabilistic aggregation controller. For more detailed surveys, including online evolution for single and multi-robot systems, see Bredeche et al. (2018); Francesca and Birattari (2016).

Embodied evolution still faces several challenges in the context of the design of control software for robot swarms. For example, robots need to execute not only their own control software but also the design process. This may not be feasible for robots with limited computational hardware. Furthermore, to conduct the evolutionary process, the swarm must operate for a relatively long time (as compared to the normal mission duration), posing more demand on batteries and increasing the likelihood of sensor or actuator failures. Additionally, without further safety measures implemented *a priori*, the robots risk to damage themselves or the environment, especially in early parts of the design process. More importantly, the evolutionary process can only be achieved if the individuals of the swarm can assess the performance of their chosen genome—ideally this should be computed for the whole swarm, however, without further infrastructure this information is not directly available to the robots as they rely only on local perception.

Three main solutions have been proposed to address the aforementioned issue: open-ended evolution, decomposition and simulation-based assessment. In open-ended evolution, the design process is not driven by an explicit objective function. Instead, open-ended evolution ties the survival of an instance of control software to its ability to "reproduce." Over time, instances of control software that successfully reproduce will replace instances of control software that cannot. Implicit selection pressure can be exerted by tying the chance to reproduce to certain desired actions or outcomes (Bianco and Nolfi, 2004; Prieto et al., 2010; Bredeche et al., 2012). For example, Bianco and Nolfi consider encounters between robots as opportunity for reproduction. As the task is self-assembly, this implicitly rewards instances of control software that manage to encounter and assemble with other robots. Another typical choice is to model the performance of the individual robots by energy levels: taking an action depletes the energy, but certain outcomes of the actions replenish it. While the robot is active (with available energy), its control software is periodically exchanged with neighboring robots. Once the energy is fully depleted, the instance of control software that was active in the robot is replaced by another one. Over time, instances of control software that are more successful at managing their energy level (by achieving the desired outcomes) will have more opportunities to spread to other robots, thus prevailing in the swarm and displacing less successful instances of control software. Like novelty search, open-ended evolution does not necessarily aim to generate a particular behavior, but rather for the spontaneous emergence of complex behaviors. As an alternative, a designer could manually decompose the objective function for the desired collective behavior into rewards for the actions (or their outcomes) of individual robots (Silva et al., 2015). Although viable, this decomposition is especially difficult in tasks that strictly require cooperation or only provide delayed rewards—e.g., taking an action does not immediately increase the fitness of the swarm, as it requires an appropriate second subsequent action to effectively increase the fitness. This decomposition is similar to the credit assignment problem encountered in robot learning (see **Section 4**). Recently, Jones et al. (2019) proposed an online evolutionary method in which robots performed simulations to evaluate the quality of genomes. This method allows the robots to estimate the performance of a genome as if it was deployed to the whole swarm—without the need for decomposing the objective function. However, assessing the performance in simulation might overestimate the degree of

cooperation and coordination of the robots, as other members of the swarm might execute different instances of control software and not cooperate as expected.

4 Robot learning

Reinforcement learning is a method for producing control software in which an agent attempts to learn a policy that encodes the set of optimal actions in a dynamic environment (Kaelbling et al., 1996). Classically, reinforcement learning only considers a single agent interacting with the environment. In this case, the system is then often modelled as a Markov decision process. As robot swarms are composed of several individuals, they are usually modelled as *multi-agent reinforcement learning* problems (see **Section 4.1**). Robot learning in swarm robotics faces similar challenges as robot evolution. Namely, *reward shaping*, the problem of specifying an appropriate reward function to generate the desired behavior, and the *reality gap*, the drop in performance observed when the control software is designed in simulation and assessed in reality. In multi-agent reinforcement learning methods, all members of the swarm typically act independently. Therefore, they are often affected by the *curse of dimensionality*, where the action space grows with the number of robots and the degrees of freedom of each robot. A variant of reinforcement learning that has found recent application in swarm robotics is *imitation learning* (see Section 4.2). Instead of optimizing the rewards gained from a known reward function, imitation learning aims to imitate a demonstrated behavior.

For reviews of robot learning in the single and multi-robot domain, see Kober et al. (2013); Zhao et al. (2020).

4.1 Multi-agent reinforcement learning

Multi-agent reinforcement learning: Robots are controlled by an instance of control software in an arbitrary form. A reinforcement learning algorithm is used to optimize the instance of control software according to a mission-specific reward function. The design process results in a single well-performing instance of control software.

Although multi-agent reinforcement learning has been largely studied in the literature, it has seen little application in swarm robotics so far. The first application of reinforcement learning in a swarm robotics scenario is possibly the one of Matarić. Matarić studied reinforcement learning with a swarm of 4 robots that perform a foraging mission (Matarić, 1997). In a follow-up work, Matarić introduced robot communication in the swarm to synchronize rewards between the robots (Matarić, 1998). More recently, Hüttenrauch et al. (2019) used deep reinforcement learning to generate control software for a swarm of virtual agents. Bloom et al. (2022) investigated the use of four deep reinforcement learning techniques in a collective transport experiment.

The application of multi-agent reinforcement learning poses several challenges that still hinder its application in swarm robotics. A first challenge arises from the fact that, in swarm robotics, the desired behavior is usually expressed at the collective level, whereas the learning must happen at the individual level. Thus, when designing control software using reinforcement learning, the mission designer needs to decompose the reward function of the whole swarm into rewards that can be assigned for individual contributions. This problem is also known as *spatial credit assignment*. To this date, no generally applicable methodology exists to address this problem and most works use manual credit assignment (Matarić, 1998; Hüttenrauch et al., 2019; Bloom et al., 2022).

Another important issue is the representation of the state and action spaces in the learning process. Typically, a multi-agent reinforcement learning uses joint action and state spaces, which are concatenated over the individual action and state spaces of each individual agent. These joint spaces, however, suffer heavily from the curse of dimensionality, as they scale poorly both in the size of the individual spaces and in the number of agents. Consequently, addressing large swarm sizes is infeasible in practice. Furthermore, the joint space is not observable by any individual agent, due to the locality of information in a robot swarm. In this sense, the problem of multi-agent reinforcement learning for swarms is more correctly modelled by a partially observable Markov decision process (Kaelbling et al., 1996). In the literature, two techniques have been mostly used to overcome the partial observability: reducing the joint action and state space to those that are pertinent to a single robot (Hüttenrauch et al., 2019; Bloom et al., 2022); or sharing information to synchronize the state beliefs of all members of the swarm (Matarić, 1998). In the first technique, a robot has no model of the behavior of its peers and they are assumed to be part of the environment. The environment that a robot experiences is therefore non-stationary; the state transitions depend not only on the actions of the individual robot but also the (changing, due to learning) behavior of its peers. In the second technique, the swarm retains some level of homogeneity by sharing information between robots. Thus, the learning process does not run independently for each robot but requires some mechanism for synchronization. When using simulations, the centralized-learning/decentralized-execution approach can be used (Hüttenrauch et al., 2019; Bloom et al., 2022). In this approach, observations of all robots are collected centrally and used to update the policy during the learning phase. However, the policy is executed decentralized on each individual robot.

4.2 Imitation learning

Imitation learning: Given demonstrations of a desired behavior, an algorithm generates an instance of control software that performs the desired behavior. Different techniques, such as behavior cloning, Turing learning, and imitation learning, have been proposed to imitate the demonstrated behavior.

A research field in reinforcement learning that has become of interest for swarm robotics researchers is imitation learning (Osa et al., 2018). In imitation learning, the reward function is assumed to be unknown. Instead, the learning process has access to demonstrations of the desired behavior. The agents attempt to learn a policy that results in a behavior that is similar to the behavior that

has been provided in a demonstration. By its own nature, imitation learning methods face a similar challenge as those of novelty search (see **Section 2.3**). Namely, how to represent a behavior numerically and how to quantitatively measure the similarity of two behaviors.

> *Turing learning*: The robots are controlled by an artificial neural network. Demonstrations are used to learn another artificial neural network that discriminates between the demonstrated behavior and the generated ones. The design process iterates between generating an instance of control software and updating the discriminator. New instances of control software are generated in such a way that they can fool the discriminator. Afterwards, the discriminator is updated to once more correctly distinguish between the demonstrated behavior and previously generated ones. The design process results in a single instance of control software, that is behaviorally similar to the demonstrated behavior, and the learned discriminator.

Within the research on imitation learning and swarm robotics, Li et al. (2016) proposed Turing learning for swarm systems. Inspired by the Turing test, the system learns two programs. A first program controls the robots in the swarm, whereas the second program attempts to distinguish between trajectories from the originally demonstrated behavior and trajectories from the behaviors that are being generated through learning.

> *Behavior cloning*: Robots are controlled by an instance of control software in an arbitrary form. Demonstrations are used to learn an instance of control software that, under the same conditions, behaves the same as the demonstrated behavior. The reward function computes the similarity of the generated behavior with regard to the demonstrated one. The design process results in a single instance of control software that closely reproduces the demonstrated behavior.

Alharthi et al. (2022) used video recordings of simulated robots to learn a behavior tree corresponding to the demonstrated collective behavior. The authors measure several swarm-level metrics, such as center of mass or length of communication paths in the swarm, and use the Jaccard distance to compute similarity with the original behavior.

> *Inverse reinforcement learning*: Robots are controlled by an instance of control software in an arbitrary form. Demonstrations are used to learn the reward function that would be maximized by the demonstrated behavior. The design process results in the learned reward function, which can then be used to learn an instance of control software for the robots.

Šošić et al. (2017) used inverse reinforcement learning to learn the behavior of two predefined particle models. Using SwarmMDP (a variant of decentralized, partially observable Markov decision processes), they reduced the multi-agent reinforcement learning problem to a single-agent problem. Gharbi et al. (2023) used apprenticeship learning (Abbeel and Ng, 2004) to learn collective behaviors from demonstrations of desired spatial organizations for a swarm.

Few studies have focused on the application of imitation learning in swarm robotics. Methods that are being currently developed face two challenges. The first challenge is that existing methods typically require detailed demonstrations to produce their corresponding control software. The more detailed the demonstrations, the easier it is to imitate them. Most work on imitation learning in swarm robotics uses an already available behavior to generate trajectories that must be learned again by the swarm (Li et al., 2016; Šošić et al., 2017; Alharthi et al., 2022). The obvious drawback of this approach is that it is only suitable for cases in which an implementation of the desired collective behavior already exists. Alternatively, other approaches have focused on only demonstrating a few key elements of the collective behavior, instead of a full trajectory (Gharbi et al., 2023). The second challenge is that there is no well-established method to measure the similarity between a demonstrated behavior and a generated one. As in novelty search (see **Section 2.3**), a collective behavior can be described by several possible forms of representations; with both characteristics at the collective and local level. The definition of characteristics at the collective level requires less domain-specific expertise to decide on, but their mathematical formulation is challenging. Individual characteristics are comparably simpler to compute, yet decomposing the desired collective behavior into its individual parts requires prior knowledge of the mission at hand.

As discussed in this section, few studies in swarm robotics have considered robot learning but have shown it to be a viable alternative to robot evolution for the design of robot swarms. Among these studies, multi-agent reinforcement learning aims to learn policies for a given reward function. Yet, like robot evolution, it faces two major challenges: reward shaping (the corresponding problem to fitness engineering) and the reality gap. Imitation learning, conversely, produces control software by imitating a demonstrated desired behavior without the reward function being known. However, it faces a similar challenge as novelty search; it relies on computing behavioral similarity (as opposed to behavioral novelty in novelty search).

5 Perspectives

As highlighted in the previous sections, the research on the design of control software for robot swarms has resulted in many promising automatic design methods. Yet, several important challenges remain. In this section, I discuss the issues that affect broad categories of automatic design methods. Additionally, I give an outlook on techniques and methods that I believe could become useful in addressing the challenges in the automatic design of control software for robot swarms.

How can we develop design methods that are robust to the reality gap?

While offline design has shown many promising results, a major challenge remains in the issue of the reality gap. Several approaches have been proposed to reduce the effects of the reality gap. For example, *system identification* can be used to develop more realistic simulators that intend to minimize the differences between simulation and reality (Bongard and Lipson, 2004; Zhao et al., 2020). However, this approach is unlikely to succeed (Jakobi, 1997), especially in swarm robotics. First, a simulation can never be identical to the system that is being simulated (Jakobi, 1997). Although investing more resources to make high-fidelity

simulations more accurate can indeed reduce the effects of the reality gap to some extent. Performing these high-fidelity simulations for up to thousands (or possibly more) individual robots would require more computing power than can reasonably be provided at the moment. Second, the robots in a swarm are relatively simple and some fault or inaccuracy in their sensors and actuators is acceptable, or even expected. Consequently, if the simulation accurately models the inaccuracies of each robot, the robots of the swarm would not be longer interchangeable: different robots are modeled to behave differently under the same circumstances. This assumption would negatively impact the desirable properties of a robot swarm. In a more general sense, research has shown that the reality gap does not affect all design methods equally and can lead to rank inversions across design methods. Indeed, a design method can perform better in simulation but worse in reality when compared to another design method (Ligot and Birattari, 2020). Therefore, I contend the focus should be on developing design methods that are inherently robust to the reality gap.

In evolutionary swarm robotics, it is often assumed that no or very little domain knowledge exists. Consequently, control software is commonly generated from scratch. However, in many tasks, there exists a reasonable amount of prior domain knowledge that could be used to ease the design process. Automatic modular design allows to incorporate this domain knowledge in the form of software modules (Birattari et al., 2021). Furthermore, the transferability of individual modules can be tested during the conception of the design method. However, the correct introduction of prior domain knowledge remains dependent on the expertise of the designer. Future research will therefore need to consider automatic modular design methods in which the modules are conceived in a mission-agnostic way; thus reducing the dependency on mission-specific domain knowledge.

When no domain knowledge is available *a priori*, other methods could be used. For example, other possible approach could be to periodically assess the transferability of the generated control software during the design process (Koos et al., 2013). The design method will therefore solve a multi-objective optimization problem, in which both the performance in simulation and in reality are considered. Assessing the control software in reality is expensive in comparison to the assessment performed in simulation. Therefore, such a design method would need to also select which instances of control software are assessed on physical robots. An often-made assumption is that control software that results in similar behaviors in simulation will transfer similarly well into reality. Starting from this assumption, one could possibly restrict the assessment of control software on real robots to instances that are behaviorally novel[2]. Ideally, the assessment should be further limited to promising solutions that have the potential to perform well in reality. However, the overly strict application of this idea might result in design methods that risk overlooking solutions that perform worse in simulation but transfer well into reality.

A further improvement could be to include a secondary simulation context (*pseudo-reality*) to assess the transferability of the control software. Ligot and Birattari have shown that the effects of the reality gap can be reproduced in simulation-only experiments (Ligot and Birattari, 2020). If combined with other transferability techniques, pseudo-reality could offer an inexpensive context to quickly assess the transferability of many instances of control software. Periodically, some instances of control software are assessed on real robots to validate the results of the pseudo-reality context and to refine it, if necessary.

How can we create online design methods for robot swarms?

Online design does not face the reality gap problem, which makes it an interesting research direction. However, the online design of robot swarms still faces several challenges. Most notably, the robots in the swarm require a way to assess their own performance solely on the local information available to them.

Another important challenge is how to avoid endangering the robots while they are interacting in an unknown environment, without the need to implement safety features *a priori*. Ideally, one could imagine that a rather general baseline behavior (or a set of baseline behaviors) is designed offline in simulation. Once the swarm is deployed in the mission environment, the swarm would then use the baseline behavior as a starting point to design its control software for the specific mission. In the simplest case, the swarm might perform a tuning of the parameters of the control software to counteract the reality gap. In more advanced scenarios, the robots might choose and combine from a set of baseline control software to find an appropriate behavior for the mission on-the-fly, and then fine-tune the parameter of the resulting control software.

How can we design control software for complex missions?

While robot swarms have already been successfully employed in a wide variety of common abstract missions, these missions usually are too simple when compared to real-world applications. More complex missions could include those with multiple, possibly conflicting objectives, or missions with dynamic environments. It is well understood that the shaping of reward and objective functions is critical, as the design process is likely to exploit any unintended local optima of the objective function (Divband Soorati and Hamann, 2015; Silva et al., 2016). Further study is required to develop engineering techniques and patterns to define appropriate objective functions.

Alternatively, incremental evolution (Gomez and Miikkulainen, 1997) and curriculum learning (Bengio et al., 2009) might find application in swarm robotics. In these approaches, a complex task is decomposed into simpler ones. The design process designs control software in increasingly complex tasks, using the previously found instances of control software as starting points for the following designs.

Orthogonal to previously mentioned approaches is (cooperative) co-evolution (Nolfi and Floreano, 1998), in which multiple subgroups of robots are evolved independently of each other to cooperate or compete in the same environment. While it is not an approach directly applicable to homogeneous systems, this approach could be beneficial for the design of heterogeneous robot swarms.

How can we design control software from other mission specifications than objective functions?

As mentioned before, the choice of an objective function is not straightforward. With the problems of bootstrapping and deception present, the designer of an objective function

2 Similar considerations as for novelty search and imitation learning—regarding what metrics can be used to characterize a behavior—apply also to this criterion.

requires not only domain knowledge of the task at hand, but also expertise in modeling collective behaviors as mathematical functions.

Novelty search has shown promising results to overcome some of these issues, however, it might lack some of the domain knowledge that can be introduced through the objective function, and that is required to obtain good performing control software. Quality-diversity algorithms combine the search for well-performing solutions commonly found in robot evolution with the exploratory search of novelty search (Mouret and Clune, 2015). Other approaches could include open-ended evolution or open-ended learning to generate varieties of different sophisticated swarm behaviors (Stanley et al., 2017; Packard et al., 2019).

Looking beyond swarm robotics, several different approaches in machine learning and evolutionary robotics have moved beyond mission-specific objective functions. For example, large language models are trained to predict the probability that a certain symbol follows the previous sequence of symbols, in an effort to imitate the texts encountered in the training set (Radford et al., 2019; Brown et al., 2020)[3]. Notably, language models are not trained on other desirable objectives except the imitation, such as grammatical correctness, reading comprehension, or trivia knowledge, yet perform well on benchmarks evaluating such objectives (Brown et al., 2020). In the context of swarm robotics, imitating examples obtained from nature might be also a viable approach to designing collective behaviors. Early works in the field were inspired by swarms in nature and often aimed at engineering artificial swarms that behaved similarly. Using imitation learning, it could be possible to automatically learn collective behaviors from swarms found in nature. Additionally, in single robot systems, another research direction makes use of demonstrations provided by human teachers (Osa et al., 2018; Krishnan et al., 2019). If these notions are applied to swarm robotics, a human teacher could demonstrate a desired collective behavior that is then used to learn the individual behavior of the robots.

6 Conclusion

The design problem in swarm robotics arises from the complexity of predicting the numerous interactions of the robots at design time. Automatic design of control software has shown to be a promising approach to tackle the design problem. However, it faces two major challenges: overcoming the reality gap and engineering appropriate objective functions. In this work, I first presented recent advances in the automatic design of control software for robot swarms. After, I discussed shortcomings of proposed approaches and provided perspectives on how to possibly overcome those.

3 In the context of language models, a symbol is not necessarily only a character but could also be a representation of any other syntactic element, such as words.

Author contributions

JK performed the review and wrote the manuscript.

Funding

The project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (DEMIURGE Project, grant agreement No 681872); from Belgium's Wallonia-Brussels Federation through the ARC Advanced Project GbO–Guaranteed by Optimization; and from the Belgian Fonds de la Recherche Scientifique–FNRS *via* the crédit d'équippement SwarmSim. Jonas Kuckling acknowledges support from the Belgian Fonds de la Recherche Scientifique–FNRS.

Acknowledgments

I would like to thank Mauro Birattari, David Garzón Ramos, Guillermo Legarda Herranz, and Ilyes Gharbi for their valuable discussions, which have shaped some of the ideas presented here.

Conflict of interest

The author declares that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

References

Abbeel, P., and Ng, A. Y. (2004). "Apprenticeship learning via inverse reinforcement learning," in *Icml 2004*. Editor C. Brodley (New York, NY, USA: ACM). doi:10.1145/1015330.1015430

Alharthi, K., Abdallah, Z. S., and Hauert, S. (2022). "Understandable controller extraction from video observations of swarms," in Swarm intelligence: 13th international conference, ANTS 2022. Editors M. Dorigo, H. Hamann, M. López-Ibáñez, J. García-Nieto, A. Engelbrecht, C. Pinciroli, et al. (Cham, Switzerland: Springer), 41–53. doi:10.1007/978-3-031-20176-9_4

Beal, J., Dulman, S., Usbeck, K., Viroli, M., and Correll, N. (2012). "Organizing the aggregate: Languages for spatial computing," in *Formal and practical aspects of domain-specific languages: Recent developments*. Editor M. Marjan (Hershey, PA, USA: IGI Global), 436–501. doi:10.4018/978-1-4666-2092-6.ch016

Beckers, R., Holland, O. E., and Deneubourg, J.-L. (1994). "From local actions to global tasks: Stigmergy and collective robotics," in Artificial life IV: Proceedings of the fourth international workshop on the synthesis and simulation of living systems. Editors R. A. Brooks, and P. Maes (Cambridge, MA, USA: MIT Press), 181–189. doi:10.7551/mitpress/1428.003.0022

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. (2009). "Curriculum learning," in ICML'09 proceedings of the 26th annual international conference on machine learning (New York, NY, USA: ACM), 41–48. doi:10.1145/1553374.1553380

Beni, G. (2005). "From swarm intelligence to swarm robotics," in Swarm robotics: SAB 2004 international workshop. Editors E. Şahin, and W. M. Spears (Berlin, Germany: Springer), 1–9. doi:10.1007/978-3-540-30552-1_1

Berman, S., Halász, Á. M., Hsieh, M. A., and Kumar, V. (2009). Optimized stochastic policies for task allocation in swarms of robots. *IEEE Trans. Robotics* 25, 927–937. doi:10.1109/TRO.2009.2024997

Berman, S., Kumar, V., and Nagpal, R. (2011). "Design of control policies for spatially inhomogeneous robot swarms with application to commercial pollination," in 2011 IEEE international conference on robotics and automation (ICRA) (Piscataway, NJ, USA: IEEE), 378–385. doi:10.1109/ICRA.2011.5980440

Bianco, R., and Nolfi, S. (2004). Toward open-ended evolutionary robotics: Evolving elementary robotic units able to self-assemble and self-reproduce. *Connect. Sci.* 16, 227–248. doi:10.1080/09540090412331314759

Birattari, M., Ligot, A., Bozhinoski, D., Brambilla, M., Francesca, G., Garattoni, L., et al. (2019). Automatic off-line design of robot swarms: A manifesto. *Front. Robotics AI* 6, 59. doi:10.3389/frobt.2019.00059

Birattari, M., Ligot, A., and Francesca, G. (2021). "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*. Editors N. Pillay, and R. Qu (Cham, Switzerland: Springer), 73–90. doi:10.1007/978-3-030-72069-8_5

Birattari, M., Ligot, A., and Hasselmann, K. (2020). Disentangling automatic and semi-automatic approaches to the optimization-based design of control software for robot swarms. *Nat. Mach. Intell.* 2, 494–499. doi:10.1038/s42256-020-0215-0

Bloom, J., Mukherjee, A., and Pinciroli, C. (2022). A study of reinforcement learning algorithms for aggregates of minimalistic robots. Available at: https://arxiv.org/abs/2203.15129.

Bongard, J. C. (2013). Evolutionary robotics. *Commun. ACM* 56, 74–83. doi:10.1145/2493883

Bongard, J. C., and Lipson, H. (2004). "Once more unto the breach: Co-evolving a robot and its simulator," in Artificial life IX: Proceedings of the ninth international conference on the simulation and synthesis of living systems. Editors J. B. Pollack, M. A. Bedau, P. Husbands, R. A. Watson, and T. Ikegami (Cambridge, MA, USA: MIT Press), 57–62. doi:10.7551/mitpress/1429.003.0011

Brambilla, M., Brutschy, A., Dorigo, M., and Birattari, M. (2014). Property-driven design for swarm robotics: A design method based on prescriptive modeling and model checking. *ACM Trans. Aut. Adapt. Syst.* 9, 1–28. doi:10.1145/2700318

Brambilla, M., Ferrante, E., Birattari, M., and Dorigo, M. (2013). Swarm robotics: A review from the swarm engineering perspective. *Swarm Intell.* 7, 1–41. doi:10.1007/s11721-012-0075-2

Bredeche, N., and Fontbonne, N. (2021). Social learning in swarm robotics. *Philosophical Trans. R. Soc. Lond. Ser. B Biol. Sci.* 377, 20200309. doi:10.1098/rstb.2020.0309

Bredeche, N., Haasdijk, E., and Prieto, A. (2018). Embodied evolution in collective robotics: A review. *Front. Robotics AI* 5, 12. doi:10.3389/frobt.2018.00012

Bredeche, N., Montanier, J.-M., Liu, W., and Winfield, A. (2012). Environment-driven distributed evolutionary adaptation in a population of autonomous robotic agents. *Math. Comput. Model. Dyn. Syst.* 18, 101–129. doi:10.1080/13873954.2011.601425

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., et al. (2020). "Language models are few-shot learners," in Advances in neural information processing systems 33 (NeurIPS 2020) (Vancouver, Canada: Curran Associates, Inc.), 1877–1901.

Cambier, N., Albani, D., Frémont, V., Trianni, V., and Ferrante, E. (2021). Cultural evolution of probabilistic aggregation in synthetic swarms. *Appl. Soft Comput.* 113, 108010. doi:10.1016/j.asoc.2021.108010

Cambier, N., and Ferrante, E. (2022). "AutoMoDe-pomodoro: An evolutionary class of modular designs," in GECCO'22: Proceedings of the genetic and evolutionary computation conference. Editor J. E. Fieldsend (New York, NY, USA: ACM), 100–103. doi:10.1145/3520304.3529031

Carrillo-Zapata, D., Milner, E., Hird, J., Tzoumas, P. J., GeorgiosVardanegaSooriyabandara, M., Giuliani, M., et al. (2020). Mutual shaping in swarm robotics: User studies in fire and rescue, storage organization, and bridge inspection. *Front. Robotics AI* 7, 53. doi:10.3389/frobt.2020.00053

Colledanchise, M., and Ögren, P. (2018). "Behavior trees in robotics and AI: An introduction," in *Chapman & Hall/CRC artificial intelligence and robotics series*. first edn (Boca Raton, FL, USA: CRC Press). doi:10.1201/9780429489105

Divband Soorati, M., and Hamann, H. (2015). "The effect of fitness function design on performance in evolutionary robotics: The influence of a priori knowledge," in GECCO'15: Proceedings of the 2015 annual conference on genetic and evolutionary computation. Editor S. Silva (New York, NY, USA: ACM), 153–160. doi:10.1145/2739480.2754676

Doncieux, S., Bredeche, N., Mouret, J.-B., and Eiben, A. (2015). Evolutionary robotics: What, why, and where to. *Front. Robotics AI* 2, 4. doi:10.3389/frobt.2015.00004

Doncieux, S., Mouret, J.-B., Bredeche, N., and Padois, V. (2011). "Evolutionary robotics: Exploring new horizons," in New horizons in evolutionary robotics: Extended contributions from the 2009 EvoDeRob Workshop. Editors J.-B. M. Stéphane Doncieux, and Nicolas Bredèche (Berlin, Germany: Springer), 1055–1062. doi:10.1007/978-3-642-18272-3_1

Dorigo, M., Birattari, M., and Brambilla, M. (2014). Swarm robotics. *Scholarpedia* 9, 1463. doi:10.4249/scholarpedia.1463

Dorigo, M., Floreano, D., Gambardella, L. M., Mondada, F., Nolfi, S., Baaboura, T., et al. (2013). Swarmanoid: A novel concept for the study of heterogeneous robotic swarms. *IEEE Robotics Automation Mag.* 20, 60–71. doi:10.1109/MRA.2013.2252996

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

Dorigo, M., Trianni, V., Şahin, E., Groß, R., Labella, H., ThomasBaldassarre, G., et al. (2003). Evolving self-organizing behaviors for a Swarm-bot. *Aut. Robots* 17, 223–245. doi:10.1023/B:AURO.0000033973.24945.f3

Dosieah, G. Y., Özdemir, A., Gauci, M., and Groß, R. (2022). "Moving mixtures of active and passive elements with robots that do not compute," in *Ants 2022: Swarm intelligence* (Cham, Switzerland: Springer), 183–195. doi:10.1007/978-3-031-20176-9_15

Duarte, M., Costa, V., Gomes, J., Rodrigues, T., Silva, F., Oliveira, S. M., et al. (2016). Evolution of collective behaviors for a real swarm of aquatic surface robots. *PLOS ONE* 11, e0151834. doi:10.1371/journal.pone.0151834

Duarte, M., Oliveira, S. M., and Christensen, A. L. (2014). "Hybrid control for large swarms of aquatic drones," in Alife 14: The fourteenth international conference on the synthesis and simulation of living systems. Editors H. Sayama, J. Rieffel, S. Risi, R. Doursat, and H. Lipson (Cambridge, MA, USA: MIT Press), 785–792. doi:10.7551/978-0-262-32621-6-ch105

Ferrante, E., Turgut, A. E., Duéñez-Guzmán, E. A., Dorigo, M., and Wenseleers, T. (2015). Evolution of self-organized task specialization in robot swarms. *PLOS Comput. Biol.* 11, e1004273. doi:10.1371/journal.pcbi.1004273

Francesca, G., and Birattari, M. (2016). Automatic design of robot swarms: Achievements and challenges. *Front. Robotics AI* 3, 1–9. doi:10.3389/frobt.2016.00029

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

Friston, K. (2010). The free-energy principle: A unified brain theory? *Nat. Rev. Neurosci.* 11, 127–138. doi:10.1038/nrn2787

Garattoni, L., and Birattari, M. (2018). Autonomous task sequencing in a robot swarm. *Sci. Robotics* 3, eaat0430. doi:10.1126/scirobotics.aat0430

Garzón Ramos, D., and Birattari, M. (2020). Automatic design of collective behaviors for robots that can display and perceive colors. *Appl. Sci.* 10, 4654. doi:10.3390/app10134654

Gauci, M., Chen, J., Dodd, T. J., and Groß, R. (2014a). "Evolving aggregation behaviors in multi-robot systems with binary sensors," in Distributed autonomous robotic systems: The 11th international symposium. Editors M. A. Hsieh, and G. Chirikjian (Berlin, Germany: Springer), 355–367. doi:10.1007/978-3-642-55146-8_25

Gauci, M., Chen, J., Li, W., Dodd, T. J., and Groß, R. (2014b). "Clustering objects with robots that do not compute," in Aamas '14: Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems (Richland, SC, USA: International Foundation for Autonomous Agents and Multiagent Systems), 421–428.

Gharbi, I., Kuckling, J., Ramos, D. G., and Birattari, M. (2023). Show me what you want: Inverse reinforcement learning to automatically design robot swarms by demonstration. Submitted for publication.

Glasmachers, T., Schaul, T., Yi, S., Wierstra, D., and Schmidhuber, J. (2010). "Exponential natural evolution strategies," in GECCO'10: Proceedings of the 12th annual conference on Genetic and evolutionary computation (New York, NY, USA: ACM), 393–400. doi:10.1145/1830483.1830557

Gomes, J., and Christensen, A. L. (2018). "Task-agnostic evolution of diverse repertoires of swarm behaviours," in Swarm intelligence: 11th international conference, ANTS 2018. Editors M. Dorigo, M. Birattari, C. Blum, A. L. Christensen, A. Reina, and V. Trianni (Cham, Switzerland: Springer), 225–238. doi:10.1007/978-3-030-00533-7_18

Gomes, J., Mariano, P., and Christensen, A. L. (2019). Challenges in cooperative coevolution of physically heterogeneous robot teams. *Nat. Comput.* 18, 29–46. doi:10.1007/s11047-016-9582-1

Gomes, J., Mariano, P., and Christensen, A. L. (2017). Novelty-driven cooperative coevolution. *Evol. Comput.* 25, 275–307. doi:10.1162/EVCO_a_00173

Gomes, J., Mariano, P., and Christensen, A. L. (2014). "Systematic derivation of behaviour characterisations in evolutionary robotics," in Alife 14: The fourteenth international conference on the synthesis and simulation of living systems (Cambridge, MA, USA: MIT Press), 212–219. doi:10.7551/978-0-262-32621-6-ch036

Gomes, J., Urbano, P., and Christensen, A. L. (2013). Evolution of swarm robotics systems with novelty search. *Swarm Intell.* 7, 115–144. doi:10.1007/s11721-013-0081-z

Gomez, F., and Miikkulainen, R. (1997). Incremental evolution of complex general behavior. *Adapt. Behav.* 5, 317–342. doi:10.1177/105971239700500305

Halloy, J., Sempo, G., Caprari, G., Rivault, C., Asadpour, M., Tâche, F., et al. (2007). Social integration of robots into groups of cockroaches to control self-organized choices. *Science* 318, 1155–1158. doi:10.1126/science.1144259

Hamann, H. (2014). "Evolution of collective behaviors by minimizing surprise," in Alife 14: The fourteenth international conference on the synthesis and simulation of living systems. Editors H. Sayama, J. Rieffel, S. Risi, R. Doursat, and H. Lipson (Cambridge, MA, USA: MIT Press), 344–351. doi:10.1162/978-0-262-32621-6-ch055

Hamann, H. (2018). *Swarm robotics: A formal approach*. Cham, Switzerland: Springer. doi:10.1007/978-3-319-74528-2

Hamann, H., and Wörn, H. (2008). A framework of space–time continuous models for algorithm design in swarm robotics. *Swarm Intell.* 2, 209–239. doi:10.1007/s11721-008-0015-3

Hansen, N., and Ostermeier, A. (2001). Completely derandomized self-adaptation in evolution strategies. *Evol. Comput.* 9, 159–195. doi:10.1162/106365601750190398

Hasselmann, K., and Birattari, M. (2020). Modular automatic design of collective behaviors for robots endowed with local communication capabilities. *PeerJ Comput. Sci.* 6, e291. doi:10.7717/peerj-cs.291

Hasselmann, K., Ligot, A., and Birattari, M. (2023). Towards the automatic design of automatic methods for the design of robot swarms. Submitted for journal publication.

Hasselmann, K., Ligot, A., Ruddick, J., and Birattari, M. (2021). Empirical assessment and comparison of neuro-evolutionary methods for the automatic off-line design of robot swarms. *Nat. Commun.* 12, 4345. doi:10.1038/s41467-021-24642-3

Hecker, J. P., Letendre, K., Stolleis, K., Washington, D., and Moses, M. E. (2012). "Formica ex machina: Ant swarm foraging from physical to virtual and back again," in Swarm intelligence: 8th international conference, ANTS 2012. Editors M. Dorigo, M. Birattari, C. Blum, A. L. Christensen, A. P. Engelbrecht, R. Groß, et al. (Berlin, Germany: Springer), 252–259. doi:10.1007/978-3-642-32650-9_25

Heinerman, J., Rango, M., and Eiben, A. (2015). "Evolution, individual learning, and social learning in a swarm of real robots," in 2015 IEEE symposium series on computational intelligence, SSCI 2015 (Los Alamitos, CA, USA: IEEE Computer Society), 1055–1062. doi:10.1109/SSCI.2015.152

Husbands, P., and Harvey, I. (1992). "Evolution versus design: Controlling autonomous robots," in Proceedings of the third annual conference of AI, simulation, and planning in high autonomy systems 'integrating perception, planning and action' (Los Alamitos, CA, USA: IEEE Computer Society), 139–146. doi:10.1109/AIHAS.1992.636878

Hüttenrauch, M., Šošić, A., and Neumann, G. (2019). Deep reinforcement learning for swarm systems. *J. Mach. Learn. Res.* 20, 1–31.

Jakobi, N. (1997). Evolutionary robotics and the radical envelope-of-noise hypothesis. *Adapt. Behav.* 6, 325–368. doi:10.1177/105971239700600205

Jones, S., Studley, M., Hauert, S., and Winfield, A. (2018). "Evolving behaviour trees for swarm robotics," in Distributed autonomous robotic systems: The 13th international symposium. Editors R. Groß, A. Kolling, S. Berman, E. Frazzoli, A. Martinoli, F. Matsuno, et al. (Cham, Switzerland: Springer), 487–501. doi:10.1007/978-3-319-73008-0_34

Jones, S., Winfield, A., Hauert, S., and Studley, M. (2019). Onboard evolution of understandable swarm behaviors. *Adv. Intell. Syst.* 1, 1900031. doi:10.1002/aisy.201900031

Kaelbling, L. P., Littman, M. L., and Moore, A. W. (1996). Reinforcement learning: A survey. *J. Artif. Intell. Res.* 4, 237–285. doi:10.1613/jair.301

Kaiser, T. K., and Hamann, H. (2019). Engineered self-organization for resilient robot self-assembly with minimal surprise. *Robotics Aut. Syst.* 122, 103293. doi:10.1016/j.robot.2019.103293

Kazadi, S. (2009). Model independence in swarm robotics. *Int. J. Intelligent Comput. Cybern.* 2, 672–694. doi:10.1108/17563780911005836

Kober, J., Bagnell, J. A., and Peters, J. (2013). Reinforcement learning in robotics: A survey. *Int. J. Robotics Res.* 32, 1238–1274. doi:10.1177/0278364913495721

Koos, S., Mouret, J.-B., and Doncieux, S. (2013). The transferability approach: Crossing the reality gap in evolutionary robotics. *IEEE Trans. Evol. Comput.* 17, 122–145. doi:10.1109/TEVC.2012.2185849

Krishnan, S., Garg, A., Liaw, R., Thananjeyan, B., Miller, L., Pokorny, F. T., et al. (2019). Swirl: A sequential windowed inverse reinforcement learning algorithm for robot tasks with delayed rewards. *Int. J. Robot. Res.* 38, 126–145. doi:10.1177/0278364918784350

Kuckling, J., Ligot, A., Bozhinoski, D., and Birattari, M. (2018). "Behavior trees as a control architecture in the automatic modular design of robot swarms," in Swarm intelligence: 11th international conference, ANTS 2018. Editors M. Dorigo, M. Birattari, C. Blum, A. L. Christensen, A. Reina, and V. Trianni (Cham, Switzerland: Springer), 30–43. doi:10.1007/978-3-030-00533-7_3

Kuckling, J., Stützle, T., and Birattari, M. (2020a). Iterative improvement in the automatic modular design of robot swarms. *PeerJ Comput. Sci.* 6, e322. doi:10.7717/peerj-cs.322

Kuckling, J., Ubeda Arriaza, K., and Birattari, M. (2020b). "AutoMoDe-IcePop: Automatic modular design of control software for robot swarms using simulated annealing," in *Artificial intelligence and machine learning: Bnaic 2019, benelearn 2019*. Editors B. Bogaerts, G. Bontempi, P. Geurts, N. Harley, B. Lebichot, T. Lenaerts, et al. (Cham, Switzerland: Springer), 3–17. doi:10.1007/978-3-030-65154-1_1

Kuckling, J., van Pelt, V., and Birattari, M. (2022). AutoMoDe-cedrata: Automatic design of behavior trees for controlling a swarm of robots with communication capabilities. *SN Comput. Sci.* 3, 136. doi:10.1007/s42979-021-00988-9

Lehman, J., and Stanley, K. O. (2011). Abandoning objectives: Evolution through the search for novelty alone. *Evol. Comput.* 19, 189–223. doi:10.1162/EVCO_a_00025

Li, S., Batra, R., Brown, D., Chang, H.-D., Ranganathan, N., Hoberman, C., et al. (2019). Particle robotics based on statistical mechanics of loosely coupled components. *Nature* 567, 361–365. doi:10.1038/s41586-019-1022-9

Li, W., Gauci, M., and Groß, R. (2016). Turing learning: A metric-free approach to inferring behavior and its application to swarms. *Swarm Intell.* 10, 211–243. doi:10.1007/s11721-016-0126-1

Ligot, A., and Birattari, M. (2020). Simulation-only experiments to mimic the effects of the reality gap in the automatic design of robot swarms. *Swarm Intell.* 14, 1–24. doi:10.1007/s11721-019-00175-w

Ligot, A., Hasselmann, K., and Birattari, M. (2020a). "AutoMoDe-arlequin: Neural networks as behavioral modules for the automatic design of probabilistic finite state machines," in Swarm intelligence: 12th international conference, ANTS 2020. Editors M. Dorigo, T. Stützle, M. J. Blesa, C. Blum, H. Hamann, M. K. Heinrich, et al. (Cham, Switzerland: Springer), 109–122. doi:10.1007/978-3-030-60376-2_21

Ligot, A., Kuckling, J., Bozhinoski, D., and Birattari, M. (2020b). Automatic modular design of robot swarms using behavior trees as a control architecture. *PeerJ Comput. Sci.* 6, e314. doi:10.7717/peerj-cs.314

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

Matarić, M. J. (1997). Reinforcement learning in the multi-robot domain. *Aut. Robots* 4, 73–83. doi:10.1023/A:1008819414322

Matarić, M. J. (1998). Using communication to reduce locality in distributed multi-agent learning. *J. Exp. Theor. Artif. Intell.* 10, 357–369. doi:10.1080/095281398146806

Mathews, N., Christensen, A. L., O'Grady, R., Mondada, F., and Dorigo, M. (2017). Mergeable nervous systems for robots. *Nat. Commun.* 8, 439. doi:10.1038/s41467-017-00109-2

Mendiburu, F. J., Garzón Ramos, D., Morais, M. R. A., Lima, A. M. N., and Birattari, M. (2022). AutoMoDe-mate: Automatic off-line design of spatially-organizing behaviors for robot swarms. *Swarm Evol. Comput.* 74, 101118. doi:10.1016/j.swevo.2022.101118

Mondada, F., Bonani, M., Raemy, X., Pugh, J., Cianci, C., Klaptocz, A., et al. (2009). "The e-puck, a robot designed for education in engineering," in Robotica 2009: Proceedings of the 9th conference on autonomous robot systems and competitions. Editors P. Gonçalves, P. Torres, and C. Alves (Castelo Branco, Portugal: Instituto Politécnico de Castelo Branco), 59–65.

Mouret, J.-B., and Clune, J. (2015). Illuminating search spaces by mapping elites. Available at: http://arxiv.org/abs/1504.04909.

Neupane, A., and Goodrich, M. (2019). "Learning swarm behaviors using grammatical evolution and behavior trees," in Proceedings of the twenty-eighth international joint conference on artificial intelligence, IJCAI-19. Editor S. Kraus (CA, USA: IJCAI Organization), 513–520. doi:10.24963/ijcai.2019/73

Nolfi, S. (2021). *Behavioral and cognitive robotics: An adaptive perspective*. Rome, Italy: Institute of Cognitive Sciences and Technologies, National Research Council.

Nolfi, S., and Floreano, D. (2000). *Evolutionary robotics: The biology, intelligence, and technology of self-organizing machines*. first edn. Cambridge, MA, USA: MIT Press.

Nolfi, S., and Floreano, D. (1998). "How co-evolution can enhance the adaptive power of artificial evolution: Implications for evolutionary robotics," in *EvoRobots 1998: Evolutionary robotics* (Berlin, Germany: Springer), 22–38. doi:10.1145/1553374.1553380

Osa, T., Pajarinen, J., Neumann, G., Bagnell, J. A., Abbeel, P., and Peters, J. (2018). An algorithmic perspective on imitation learning. *Found. Trends® Robot* 7, 1–179. doi:10.1561/2300000053

Özdemir, A., Gauci, M., Groß, R., and Gros, R. (2018). Finding consensus without computation. *IEEE Robotics Automation Lett.* 3, 1346–1353. doi:10.1109/LRA.2018.2795640

Özdemir, A., Gauci, M., and Groß, R. (2017). "Shepherding with robots that do not compute," in ECAL 2017, the fourteenth European conference on artificial life (Cambridge, MA, USA: MIT Press). doi:10.7551/ecal_a_056

Özdemir, A., Gauci, M., Kolling, A., Hall, M. D., and Groß, R. (2019). "Spatial coverage without computation," in 2019 international conference on robotics and automation (ICRA) (Piscataway, NJ, USA: IEEE), 9674–9680. doi:10.1109/ICRA.2019.8793731

Packard, N., Bedau, M. A., Channon, A., Ikegami, T., Rasmussen, S., Stanley, K. O., et al. (2019). An overview of open-ended evolution: Editorial introduction to the open-ended evolution ii special issue. *Artif. Life* 25, 93–103. doi:10.1162/artl_a_00291

Pinciroli, C., and Beltrame, G. (2016). Buzz: A programming language for robot swarms. *IEEE Softw.* 33, 97–100. doi:10.1109/MS.2016.95

Prieto, A., Becerra, J. A., Bellas, F., and Duro, R. J. (2010). Open-ended evolution as a means to self-organize heterogeneous multi-robot systems in real time. *Robotics Aut. Syst.* 58, 1282–1291. doi:10.1016/j.robot.2010.08.004

Prorok, A., Hsieh, M. A., and Kumar, V. (2009). The impact of diversity on optimal control policies for heterogeneous robot swarms. *IEEE Trans. Robotics* 33, 346–358. doi:10.1109/TRO.2016.2631593

Pugh, J. K., Soros, L. B., and Stanley, K. O. (2016). Quality diversity: A new frontier for evolutionary computation. *Front. Robotics AI* 3, 40. doi:10.3389/frobt.2016.00040

Quinn, M., Smith, L., Mayley, G., and Husbands, P. (2003). Evolving controllers for a homogeneous system of physical robots: Structured cooperation with minimal sensors. *Philos. Trans. A Math. Phys. Eng. Sci.* 361, 2321–2343. doi:10.1098/rsta.2003.1258

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners. Available at: https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

Reina, A., Valentini, G., Fernández-Oto, C., Dorigo, M., and Trianni, V. (2015). A design pattern for decentralised decision making. *PLOS ONE* 10, e0140950. doi:10.1371/journal.pone.0140950

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

Şahin, E. (2005). "Swarm robotics: From sources of inspiration to domains of application," in Swarm robotics: SAB 2004 international workshop. Editors E. Şahin, and W. M. Spears (Berlin, Germany: Springer), 10–20. doi:10.1007/978-3-540-30552-1_2

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

Schranz, M., Di Caro, G. A., Schmickl, T., Elmenreich, W., Arvin, F., Şekercioğlu, Y. A., et al. (2021). Swarm intelligence and cyber-physical systems: Concepts, challenges and future trends. *Swarm Evol. Comput.* 60, 100762. doi:10.1016/j.swevo.2020.100762

Schranz, M., Umlauft, M., Sende, M., and Elmenreich, W. (2020). Swarm robotic behaviors and current applications. *Front. Robotics AI* 7, 36. doi:10.3389/frobt.2020.00036

Silva, F., Correia, L., and Christensen, A. L. (2017). Evolutionary online behaviour learning and adaptation in real robots. *R. Soc. Open Sci.* 4, 160938. doi:10.1098/rsos.160938

Silva, F., Duarte, M., Correia, L., Oliveira, S. M., and Christensen, A. L. (2016). Open issues in evolutionary robotics. *Evol. Comput.* 24, 205–236. doi:10.1162/EVCO_a_00172

Silva, F., Urbano, P., Correia, L., and Christensen, A. L. (2015). odNEAT: an algorithm for decentralised online evolution of robotic controllers. *Evol. Comput.* 23, 421–449. doi:10.1162/EVCO_a_00141

Slavkov, I., Carrillo-Zapata, D., Carranza, N., Diego, X., Jansson, F., Kaandorp, J., et al. (2018). Morphogenesis in robot swarms. *Sci. Robotics* 3, eaau9178. doi:10.1126/scirobotics.aau9178

Šošić, A., Khuda Bukhsh, W. R., Zoubir, A. M., and Koeppl, H. (2017). "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), 1413–1421.

Soysal, O., and Şahin, E. (2007). "A macroscopic model for self-organized aggregation in swarm robotic systems," in Swarm robotics: Second international workshop, SAB 2006. Editors E. Şahin, W. M. Spears, and A. Winfield (Berlin, Germany: Springer), 27–42. doi:10.1007/978-3-540-71541-2_3

Spaey, G., Kegeleirs, M., Garzón Ramos, D., and Birattari, M. (2020). "Evaluation of alternative exploration schemes in the automatic modular design of robot swarms," in *Artificial intelligence and machine learning: Bnaic 2019, benelearn 2019*. Editors B. Bogaerts, G. Bontempi, P. Geurts, N. Harley, B. Lebichot, T. Lenaerts, et al. (Cham, Switzerland: Springer), 18–33. doi:10.1007/978-3-030-65154-1_2

Stanley, K. O., Lehman, J., and Soros, L. (2017). Open-endedness: The last grand challenge you've never heard of. Available at: https://www.oreilly.com/radar/open-endedness-the-last-grand-challenge-youve-never-heard-of/.

Stanley, K. O., and Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evol. Comput.* 10, 99–127. doi:10.1162/106365602320169811

Trianni, V. (2014). Evolutionary robotics: Model or design? *Front. Robotics AI* 1, 13. doi:10.3389/frobt.2014.00013

Trianni, V. (2008). *Evolutionary swarm robotics*. Berlin, Germany: Springer. doi:10.1007/978-3-540-77612-3

Trianni, V., Groß, R., Labella, H., ThomasŞahin, E., and Dorigo, M. (2003). "Evolving aggregation behaviors in a swarm of robots," in Advances in artificial life: 7th European conference, ECAL 2003. Editors W. Banzhaf, J. Ziegler, T. Christaller, P. Dittrich, and J. T. Kim (Berlin, Germany: Springer), 865–874. doi:10.1007/978-3-540-39432-7_93

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

Trianni, V., and Nolfi, S. (2011). Engineering the evolution of self-organizing behaviors in swarm robotics: A case study. *Artif. Life* 17, 183–202. doi:10.1162/artl_a_00031

Trianni, V., Tuci, E., Ampatzis, C., and Dorigo, M. (2014). "Evolutionary swarm robotics: A theoretical and methodological itinerary from individual neuro-controllers to collective behaviours," in *The horizons of evolutionary robotics*. Editors P. A. Vargas, E. A. D. Paolo, I. Harvey, and P. Husbands (Boston, MA: MIT Press), 153–178. doi:10.7551/mitpress/8493.003.0008

van Diggelen, F., Luo, J., Cambier, N., Ferrante, E., and Eiben, A. (2022). "Environment induced emergence of collective behavior in evolving swarms with limited sensing," in GECCO'22: Proceedings of the genetic and evolutionary computation conference. Editor J. E. Fieldsend (New York, NY, USA: ACM), 31–39. doi:10.1145/3512290.3528735

Watson, R. A., Ficici, S. G., and Pollack, J. B. (2002). Embodied evolution: Distributing an evolutionary algorithm in a population of robots. *Robotics Aut. Syst.* 39, 1–18. doi:10.1016/S0921-8890(02)00170-7

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

Winfield, A., Harper, C. J., and Nembrini, J. (2005). "Towards dependable swarms and a new discipline of swarm engineering," in Swarm robotics: SAB 2004 international workshop. Editors E. Şahin, and W. M. Spears (Berlin, Germany: Springer), 126–142. doi:10.1007/978-3-540-30552-1_11

Xie, H., Sun, M., Fan, X., Lin, Z., Chen, W., Wang, L., et al. (2019). Reconfigurable magnetic microrobot swarm: Multimode transformation, locomotion, and manipulation. *Sci. Robotics* 4, eaav8006. doi:10.1126/scirobotics.aav8006

Yamins, D., and Nagpal, R. (2008). "Automated global-to-local programming in 1-D spatial multi-agent systems," in Aamas '08: The seventh international conference on autonomous agents and multiagent systems. Editors L. Padgham, D. Parkes, J. Müller, and S. Parsons (Richland, SC, USA: International Foundation for Autonomous Agents and Multiagent Systems), 615–622.

Yang, G.-Z., Bellingham, J., Dupont, P. E., Fischer, P., Floridi, L., Full, R., et al. (2018). The grand challenges of Science Robotics. *Sci. Robotics* 3, eaar7650. doi:10.1126/scirobotics.aar7650

Yu, J., Wang, B., Du, X., Wang, Q., and Zhang, L. (2018). Ultra-extensible ribbon-like magnetic microswarm. *Nat. Commun.* 9, 3260. doi:10.1038/s41467-018-05749-6

Zhao, W., Queralta, J. P., and Westerlund, T. (2020). "Sim-to-real transfer in deep reinforcement learning for robotics: A survey," in 2020 IEEE symposium series on computational intelligence (SSCI) (Piscataway, NJ, USA: IEEE), 737–744. doi:10.1109/SSCI47803.2020.9308468