frontiers in Robotics and AI

REVIEW
published: 26 May 2016
doi: 10.3389/frobt.2016.00029

Automatic Design of Robot Swarms: Achievements and Challenges

*Gianpiero Francesca and Mauro Birattari**

IRIDIA, Université Libre de Bruxelles, Brussels, Belgium

Automatic design is a promising approach to the design of control software for robot swarms. In an automatic design method, the design problem is cast into an optimization problem and is addressed using an optimization algorithm. In this article, we review studies in which automatic design methods are successfully applied. In particular, we focus our attention on how automatic methods are empirically assessed. An apparent issue that emerges from our review is that a solid, well-established, and consistently applied empirical practice is still missing. For example, studies that propose new methods and ideas do not typically provide any comparison with existing ones. We maintain that the lack of a proper empirical practice hinders the progress of the domain. In this article, we pursue two goals: we highlight the notable achievements in the automatic design of control software for robot swarms and we discuss the challenges to be overcome to establish a proper empirical practice for the domain.

OPEN ACCESS

Edited by:
Andreas Kolling, University of Sheffield, UK

Reviewed by:
Heiko Hamann, University of Paderborn, Germany
Yuri Kaszubowski Lopes, University of Sheffield, UK

***Correspondence:**
Mauro Birattari
mbiro@ulb.ac.be

Specialty section:
This article was submitted to Multi-Robot Systems, a section of the journal Frontiers in Robotics and AI

Received: *09 February 2016*
Accepted: *02 May 2016*
Published: *26 May 2016*

Citation:
Francesca G and Birattari M (2016) Automatic Design of Robot Swarms: Achievements and Challenges. Front. Robot. AI 3:29. doi: 10.3389/frobt.2016.00029

Keywords: swarm robotics, collective robotics, automatic design, design methodology, evolutionary robotics

1. INTRODUCTION

In swarm robotics, a large number of robots are deployed to accomplish a mission that is beyond the capabilities of a single robot (Dorigo et al., 2014). Because a single robot is not able to accomplish the mission on its own, the robots must cooperate. A robot swarm operates in a self-organized and distributed manner: there is no leader and coordination is obtained via interaction between the individual robots. Moreover, a robot swarm does not rely on any external infrastructure: each individual robot acts on the basis of local information obtained through its sensors or provided by neighboring robots via local communication.

Although the literature describes a number of robot swarms that have been developed and demonstrated, a reliable engineering approach to the design of control software for robot swarms is still at dawn (Brambilla et al., 2013). Typically, designers proceed by trial and error, guided only by their intuition and experience. Some effort has been made recently to overcome this problem and a number of principled manual methods have been proposed [e.g., see Hamann and Wörn (2008), Kazadi et al. (2009), Berman et al. (2011), Werfel et al. (2014), Brambilla et al. (2015), Reina et al. (2015), and Lopes et al. (2016)]. Although interesting and promising results have been obtained, we are far from a generally applicable solution.

Automatic methods are a promising alternative. In an automatic method, the problem of designing the control software for a robot swarm is cast into an optimization problem: the different design choices define a search space that is explored using an optimization algorithm. Most of the automatic methods proposed so far belong to the framework of evolutionary robotics (Nolfi and Floreano, 2000). Traditionally in evolutionary robotics, the control software is based on artificial neural networks and is optimized automatically via an evolutionary algorithm, following a process inspired by natural

evolution. As discussed in Section 2, evolutionary robotics has been successfully adopted to design robot swarms that perform various tasks. The results achieved show that automatic design is a viable and promising approach to design the control software of robot swarms.

Unfortunately, the pioneering achievements registered so far are to be considered as somehow isolated contributions, rather than the incremental acquisitions of an established and mature science.[1] With few exceptions, studies that introduce new automatic design methods and ideas do not provide any comparison with previously introduced ones. Indeed, a solid, well-established, and consistently applied practice for the empirical assessment and comparison of automatic design methods is still missing.

We are convinced that the lack of a proper empirical practice hinders the progress of the research on the automatic design of control software for robot swarms. We contend that a well-established empirical practice that encourages comparisons would properly promote the best ideas proposed so far, would help to focus on promising directions, and would attract further researchers and investments to the domain.

With this article, we pursue two objectives. In Section 2, we highlight studies that present notable achievements in the domain of the automatic design of control software for robot swarms. We devote particular attention to the empirical approach adopted to assess new proposals. In Section 3, we discuss some challenges that the research community should overcome to transform the current research on the automatic design of control software for robot swarms into a mature science.

2. ACHIEVEMENTS

In automatic design, the problem of designing the control software is cast into an optimization problem. In other terms, an automatic design method uses an optimization algorithm to search the design space. This space comprises all the instances of control software that the method can possibly produce. The goal of the optimization algorithm is to find an instance of control software that maximizes an appropriate performance measure. Automatic design methods can be divided into two classes: *off-line* and *on-line* methods.

2.1. Off-Line Methods

In off-line methods, the design process takes place in a preliminary, dedicated phase: the design phase. The design phase occurs and terminates before the robot swarm is deployed in its operational environment. Within the design process, an off-line method evaluates a relatively large number of different instances of control software. Typically, the evaluation of an instance of control software is performed via a computer-based simulation. Simulation offers two main benefits: (i) it enables a faster-than-real-time evaluation and (ii) prevents the robots from being damaged by a possibly low-quality instance of the control software.

Evolutionary robotics (Nolfi and Floreano, 2000) is the most studied automatic design approach in swarm robotics. Typically, in evolutionary robotics an evolutionary algorithm is used to optimize the parameters and possibly the structure of a neural network that takes as an input sensor readings and returns actuation commands. Originally, evolutionary robotics was successfully applied in single robot scenarios. The adoption of the evolutionary robotics approach in swarm robotics goes under the name of *evolutionary swarm robotics* (Trianni, 2008). In the following, we present a number of notable achievements in evolutionary swarm robotics. For comprehensive reviews of the (single-robot) evolutionary robotics literature, we refer the reader to Bongard (2013), Doncieux and Mouret (2014), Trianni (2014), and Silva et al. (2016).

Most of the notable works in evolutionary swarm robotics share a set of common characteristics: (1) the swarms produced are behaviorally homogeneous, that is, all the robots of the swarm execute identical copies of the same control software. (2) The objective function that is optimized during the design process is globally and centrally evaluated, that is, it is computed considering the performance of the swarm as a whole. (3) The optimization algorithm adopted within the design process is a classical evolutionary algorithm that features elitism, recombination, and mutation. (4) The size of the population ranges from 50 to 200 individuals, with 100 individuals as the most common value. (5) The number of performance assessments for each instance of the control software ranges from 3 to 100, with 10 as the most common value. Each assessment differs from the others in the initial conditions – i.e., position of the robots and characteristics of the environment. (6) To take into account stochasticity, different independent runs of the evolutionary algorithm are performed. The most common number of independent runs is ten. (7) Unfortunately, the assessment of the obtained control software, in reality, is not always performed. Many studies present results in simulation only. When robot experiments are performed, they are often only isolated demonstrations rather than a structured empirical analysis aimed at producing statistically significant results.

Since the introduction of evolutionary swarm robotics, most of the research effort aimed at showing the feasibility of the approach and investigating whether a particular collective behavior can be obtained via artificial evolution. Quinn et al. (2003) were the first to adopt the evolutionary approach in the context of swarm robotics. The authors obtained a coordinated motion behavior by using an evolutionary algorithm to optimize control software based on a neural network. Robot experiments were performed with three Khepera robots. More than a decade after the publication of this work, some aspects of the experimental design might appear unusual. For example, the performance of each instance of control software produced within the design process was assessed via 60 evaluations. In the final stage of the design process, the number of evaluations was increased to 100. At the end of the design process, 10 different instances of control software were tested in robot experiments. The very best instance was tested through 100 runs. Following this seminal work, a number of robot swarms designed via evolutionary robotics have been described in the literature. For instance, Christensen and Dorigo (2006) showed how to use evolution to obtain a swarm of robots that is able to perform hole-avoidance and phototaxis

[1] We use the notion of *mature science* as defined by Kuhn (1962).

at the same time. The study includes a comparison between three different evolutionary algorithms: a classical evolutionary algorithm, evolutionary strategy, and a co-evolutionary genetic algorithm. For each algorithm, 20 independent runs were performed. Evolutionary strategy yielded the best performance. The best instance of control software produced by evolutionary strategy was tested in experiments performed with a group of three s-bot robots. Under a similar setting, Baldassarre et al. (2007) obtained coordinated motion with a swarm of four physically connected s-bot robots. The best instance of control software obtained across 20 independent runs of the design process was tested on a group of four s-bot robots. In the experiments performed by the authors, the control software obtained via computer-based simulations performed well in reality without the need of any adjustment. Moreover, the same control software performed well also when the robots had to navigate on rough terrains. Hauert et al. (2008) used evolutionary robotics to obtain the control software for a swarm of aerial robots that are required to establish a wireless communication network. Experiments were performed in simulation only. Trianni and Nolfi (2009) studied the design of a self-organizing behaviors via evolution. The experimental analysis was performed mostly in simulation and involved 20 independent runs of the design process. The best instance of control software obtained across the 20 runs was tested on groups of two and three s-bot robots. Waibel et al. (2009) investigated the influence on performance of different selective pressures (individual and collective) and different team compositions (homogeneous and heterogeneous swarms). The experimental analysis was conducted on three different variants of a foraging task. For each variant, four different combinations of selective pressures and team compositions were tested via 20 independent runs of the design process. Experiments were conducted both in simulation and with ten Alice robots. Ferrante et al. (2015) used a method based on grammatical evolution (Ferrante et al., 2013) to evolve task specialization in a robot swarm. In particular, the authors highlighted the environmental conditions that are necessary for task specialization to emerge. Experiments were performed in simulation only.

Recently, the research in evolutionary swarm robotics has been influenced by the current trends in evolutionary computation. The studies on *novelty search* and *multi-objective optimization* are worthy of mention. Novelty search (Lehman and Stanley, 2011) is an approach to evolutionary computation that promotes diversity instead of performance. Results indicate that novelty search is robust to issues that affect the classical evolutionary approach, including premature convergence and stagnation. Gomes et al. (2013) introduced novelty search in the context of swarm robotics. They used novelty search to automatically develop control software for aggregation and resource sharing. The study includes a comparison with a classical evolutionary approach and with a hybrid approach that combines a classical approach and novelty search. Experiments were conducted in simulation only.

Multi-objective optimization focuses on problems in which multiple, possibly conflicting objectives are to be optimized. In evolutionary robotics, multi-objective optimization might be of interest in two contexts: (i) the problem is naturally formulated as a multi-objective problem. For example, the task is composite and the robots are required to accomplish multiple sub-tasks, each associated with a performance measure (Capi, 2007). (ii) The designer wishes to promote a specific property that can be characterized via a measurable quantity. For example, the designer wishes to increase the chance that the obtained control software works properly in reality (Koos et al., 2013b), enforce a low complexity of the control software produced by the optimization process (Teo and Abbass, 2004, 2005), or increase the robustness against hardware failures and environmental variability (Koos et al., 2013a; Lehman et al., 2013). Trianni and López-Ibáñez (2015) were the first to study the application of multi-objective optimization in the context of evolutionary swarm robotics. They presented results on two swarm robotics tasks: flocking and an idealized version of the stick-pulling experiment. Experiments were conducted in simulation only.[2]

A number of notable studies depart from the classical evolutionary robotics tradition by adopting (i) control software architectures other than monolithic neural networks and/or (ii) optimization algorithms other than evolutionary computation. In particular, some authors studied the possibility of using an optimization algorithm to fine-tune a parametric control architecture that features only a small set of parameters. Hecker et al. (2012) obtained a foraging behavior by using artificial evolution to optimize the parameters of a probabilistic finite state machine. Experiments were conducted with a group of three custom-made two-wheeled robots. The objects to be retrieved were emulated via RFID tags placed on the ground. The tags could be sensed by the robots via an onboard RFID reader. Gauci et al. (2014a) obtained an object clustering behavior by using evolutionary strategy to optimize a simple control architecture that features six parameters. The simplicity of the control architecture arises from the fact that the control architecture controls the speed of the two wheels of the robot exclusively based on the readings of a single three-state *line-of-sight* sensor. The line-of-sight sensor indicates whether the robot faces (i) another robot, (ii) an object to cluster, or (iii) neither of them. On the robot, the line-of-sight sensor was implemented using the frontal camera and a color detector that recognizes robots that are green and objects that are red. Experiments were conducted with a group of five e-puck robots. Ten independent runs of the design process were performed in simulation. For each run, the best instance of control software obtained was tested in reality. Gauci et al. (2014b) optimized the same control architecture via exhaustive search to obtain self-organized aggregation. In this second study, no object is present in the environment and the line-of-sight sensor can be, therefore, in only two states: it indicates whether the robot faces (i) another robot or (ii) not. Due to the even simpler control architecture adopted in this second study, the design space counts only 194,481 different instances of control software that can be possibly generated. The reduced size of the design space allows for exhaustive search in simulation. Experiments were conducted with a group of forty e-puck robots: the best performing instance of control software found via exhaustive search was tested 30 times on the robots.

[2]The authors presented results also on a single-robot task.

Other authors proposed the adoption of a modular control architecture: the control software is obtained by combining modules obtained either via manual or automatic design. Duarte et al. (2014a) presented an approach based on the hierarchical decomposition of complex behaviors into simpler behaviors that can be generated either manually or via evolutionary robotics. The simple behaviors are then combined using high-level neural networks. The approach was illustrated in simulation only, on an object retrieval task. The results indicate that the approach outperforms the classical, monolithic evolutionary approach. Similarly, Duarte et al. (2014b) used hierarchical decomposition to obtain control software for a patrolling task. The control software comprises low-level behaviors (go to waypoint, patrol, pursue intruder) and one module that arbitrates the low-level behaviors. The low-level behaviors were implemented as neural networks and were obtained via evolution. The arbitrator was manually written. The resulting system was tested in simulation only. The downside of the approach is that hierarchical decomposition is task dependent and has to be performed manually by the designer. In a successive study, the authors designed via artificial evolution control software for a swarm of ten aquatic robots (Duarte et al., 2015; Christensen et al., 2016). The task to be performed by the swarm comprised four different sub-tasks: homing, dispersion, clustering, and area monitoring. The aggregated control software was obtained by sequentially combining the instances of control software developed for each sub-task. Experiments were performed in a 330 m × 190 m waterbody next to the Tagus river in Lisbon, Portugal. The results show that the control software produced via the proposed method crosses the reality gap nicely.

Another modular approach that has been recently proposed is AutoMoDe (Francesca et al., 2014). In AutoMoDe, the control software is a probabilistic finite state machine that is automatically synthesized by combining pre-existing, task-independent modules. In Francesca et al. (2014), Vanilla, a first implementation of AutoMoDe, was tested on two different tasks: aggregation and foraging. In a set of experiments conducted with a swarm of twenty e-puck robots, Vanilla outperformed a classical evolutionary swarm robotics method. In Francesca et al. (2015), Chocolate, a second implementation of AutoMoDe, was introduced. Chocolate differs from Vanilla in the fact that it adopts a more powerful optimization algorithm. Chocolate was compared against Vanilla and human designers on five tasks. The experimental analysis performed on twenty e-puck robots showed that the control software produced by Chocolate outperforms the one produced by Vanilla and by the human designers who participated in the study.

2.2. On-Line Methods

In on-line methods, the design process takes place when the robot swarm has been already deployed in its operational environment. The apparent advantage of on-line methods over off-line methods is that on-line methods can benefit from the availability of information on the actual operational environment that would be unavailable at off-line design time. Moreover, one could expect that an on-line method can cope with changes in operating conditions and adapt to contingencies. In other words, on-line methods aspire to produce a design that is more tailored to the specific mission than the one produced by an off-line method. On the downside, on-line methods are likely constrained to a reduced search space with respect to off-line methods because of two main reasons: (i) as the design process is performed by the robots themselves while they are operational, computational resources and time are limiting factor and (ii) candidate design that are potentially dangerous for the robots should be *a priori* removed from the search space to avoid jeopardizing the mission. Moreover, in a realistic on-line setting, the design process is fully distributed on the robots and cannot count on any centralized entity to measure performance and guide the search, as it can in an off-line setting. The design process is, therefore, constrained to use performance indicator that can be evaluated in a distributed and local way. Whether on-line methods are more or less effective than off-line methods, which are the features of a mission that are better handled by on-line or off-line methods, and whether on-line and off-line methods can be effectively combined are empirical questions that require further research to be answered convincingly.

In the works on the on-line automatic design of control software for robot swarms, we can highlight some typical characteristics: (1) each robot of the swarm explores a portion of the search space. Typically, each robot evaluates asynchronously a subpopulation of instances of control software and keeps track of their performance. (2) The robots exchange information to co-ordinate the search process. Although the way in which information is exchanged varies from work to work, a typical feature is that best performing instances of control software are shared between the robots. These instances are typically modified using mutation by the sender or by the receiver and become part of the subpopulation of instances explored by the receiver. (3) Robot swarms are *de facto* behaviorally heterogeneous, that is, at any given moment each robot executes a different instance of control software. This does not necessarily exclude that the robots can eventually converge to executing the same instance of control software. (4) As each robot of the swarm is required to evaluate instances of control software, the objective function must be computable relying only on information that is locally available to the robot. This restricts the class of tasks that are tackled in the literature (and that can be possibly tackled) using the on-line approach. The tasks that can be tackled appear to be simpler and less diverse than those that are tackled in the off-line approach. (5) Unfortunately, experiments in reality are not always performed. Some studies present results in simulation only. We can identify four different ways in which experiments are performed: a. experiments are conducted in simulation only; b. the design of the control software is conducted in simulation and then a subset of the best instances of control software are tested on the robots (during the test the robots do not perform on-line design); c. the design of the control software is conducted partly in simulation and partly on the robots. The obtained control software is tested on the robots; and d. the design of the control software is conducted exclusively on the robots. The obtained control software is tested on the robots.

One of the first studies proposing an on-line method for multi-robot systems was published by Parker (1996, 1997). The author

proposed L-ALLIANCE, a behavior-based control architecture that allows the on-line optimization of some internal parameters of the control software. No robot experiment is described in the articles in which the method is introduced. Matarić (2001) surveyed various behavior-based control architectures that allow the on-line optimization of parameters. Lee and Arkin (2003) applied learning momentum (Clark et al., 1992) to a multi-robot system. The task that they considered is an abstraction of capture-the-flag. Experiments were performed in simulation only.

A number of significant studies on on-line automatic design belong to the *embodied evolution* approach (Watson et al., 1999, 2002). In embodied evolution, an evolutionary algorithm that is in charge of optimizing the control software is executed in a distributed way. The computation is performed by the robots of the swarm while they are situated in the operational environment. Watson et al. (1999, 2002) developed a multi-robot system that is able to achieve phototaxis via embodied evolution. In this system, each robot broadcasts a mutated version of the best instance of control software it obtained, with a probability that is a function of its performance. Experiments were conducted with eight custom-made two-wheeled robots. To overcome the limited duration of the batteries and to grant a sufficient amount of time to the design process, robots were connected to a power source using special hardware that makes contact with an electrified floor. Similarly, Usui and Arita (2003) applied embodied evolution to design a simple obstacle avoidance behavior on six Khepera robots. The study includes a comparison between a group of robots that exchange the best performing instances of control software they found and a group that does not. The results indicate that exchanging control software is beneficial. As in Watson et al. (2002), the limited duration of the batteries was overcome using special hardware that provides a continuous power supply; in this case, a pantograph that makes constant contact with an overhead power line. Bianco and Nolfi (2004) proposed a method that allows robots to share their internal configuration via self-assembly. The method was tested in simulation only. König and Mostaghim (2009) departed from the classical approach based on neural networks, by proposing a control architecture based on finite state machines optimized on-line via embodied evolution. The tasks considered were gate passing and collision avoidance. The experiments were conducted in simulation only.

A few studies were devoted to the investigation of fundamental scientific questions concerning the relationship between and the combination of learning and evolution. Wischmann et al. (2007) studied the relationship between learning and embodied evolution in a predator–prey scenario. Experiments were performed in simulation only. Elfwing et al. (2011) investigated the combination of reinforcement learning and embodied evolution in a survival scenario. Experiments were conducted in simulation and the control software obtained was then evaluated on two Cyber Rodent robots. The authors followed a hybrid procedure: the design was conducted in simulation using initially a group of four robots and then a group of two. The resulting control software was eventually tested on the two Cyber Rodent robots.

More recent studies shifted the focus back to the development of effective on-line design methods. Bredeche et al. (2012) proposed MEDEA, a framework for onboard evolution of control software based on neural networks. In MEDEA, robots are able to adapt to environmental conditions. The experimental analysis was performed on two tasks (survival and two suns) using twenty e-puck robots. Some precautions were taken in the definition of the experiments: (i) to reach convergence within the duration of the batteries, experiments were conducted using a small search space; (ii) to allow the exchange of solutions between robots, a reliable local communication was emulated via Wi-Fi and information provided by a tracking system. Haasdijk et al. (2014) introduced a multi-objective variant of MEDEA called MONEE. MONEE allows a swarm of robots to adapt their behavior to the environment and to a given task at the same time. Experiments were performed in simulation only. Silva et al. (2015) introduced odNEAT, a variant of real-time NEAT (Stanley and Miikkulainen, 2002) for the online and decentralized optimization of robot control software. Experiments on three multi-robot tasks (aggregation, navigation with obstacle avoidance, and phototaxis) were performed in simulation only.

A number of studies presented on-line design methods based on distributed implementations of particle swarm optimization (Pugh and Martinoli, 2009). Here, we mention only two recent studies. We refer the reader to Di Mario et al. (2015) for a more extensive review of the relevant literature. Di Mario and Martinoli (2014) applied distributed particle swarm optimization to the onboard optimization of control software for obstacle avoidance. The method was tested on a swarm of eight Khepera III robots. The experiments were performed in three different settings: (i) the control software was designed in simulation and then tested on the robots; (ii) the control software was designed and tested on the robots; and (iii) the control software was designed partly in simulation and partly on the robots, to be eventually tested on the robots. In a similar study, Di Mario et al. (2015) presented OCBA, a noise-resistant particle swarm optimization algorithm for the on-line automatic design of robot control software. In this case, the on-line adaptation was performed in simulation. The resulting control software was then tested on a swarm of four Khepera III robots.

3. CHALLENGES

Although a number of promising methods for the automatic design of control software for robot swarms have been presented and discussed so far, a consolidated literature on the topic is still missing. In some sense, we could (somehow provocatively) argue that the state of the art in the automatic design of control software for robot swarms is undefined and is yet to be properly identified. As we have seen in Section 2, with the exception of a few rare cases, published studies do not produce comparisons between different methods. Moreover, the vast majority of the articles in which interesting automatic design methods have been introduced were tailored to answer scientific questions that do not directly belong in automatic design. For example, an important share of the articles devoted to evolutionary robotics were tailored to answer scientific questions related to the plausibility of biological models or the justification of animal behaviors in evolutionary terms. These questions are clearly extremely fascinating and relevant in absolute terms. Nonetheless, in many cases, these questions have

unfortunately shadowed the core questions of the research on the automatic design of control software for robot swarms. Indeed, some key questions are hardly addressed in the current literature: *Which automatic method is the best under which conditions? How general is method X? How well does method X perform on different tasks? Which features of a task pose problems to method X? How well does a swarm produced by method X cross the reality gap?* Questions of this nature are in our opinion fundamental for the development of a mature science. In particular, the development of a solid, well-established, and consistently applied empirical practice to assess and compare methods is of paramount importance. Within the field of artificial intelligence, domains that rightfully qualify as mature science greatly invested in the development of a proper empirical practice. Think, for example, of machine learning and heuristic optimization – for example, evolutionary computation, particle swarm optimization, and ant colony optimization. In these domains, virtually all published articles propose an extensive experimental analysis and all newly proposed ideas are thoroughly compared empirically against the established state of the art. Moreover, in these domains, it is common to share benchmark problems, datasets, implementations, and results.

It is our contention that the research on the automatic design of control software for robot swarms cannot significantly progress further unless the research community endows itself with a strong, common empirical practice. Comparisons between different design methods should play a much more prominent role. At every moment in time, the research community should be mindful of what is the best method for a given problem, what is the relative performance of the available methods, and which are the relative strengths and weaknesses of each existing method. In other words, the state of the art should be precisely defined and every new proposed method should be compared with it.

In the following, we discuss four main issues to be addressed in order to define an empirical practice that is appropriate for the automatic design of control software for robot swarms.

3.1. Reference Model

To be meaningful, a comparison of design methods must be performed under the same conditions for all methods under analysis. It appears obvious that all methods must be given the same resources: computation time, memory, simulator and simulation models, number and kind of CPUs, operating system, hardware infrastructure, etc. It appears also obvious that the different design methods under analysis must be requested to produce control software for the same robotic platform. This last requirement is less trivial than one might think. Indeed, we became convinced that simply stating which platform is considered in a study is not sufficient to guarantee that all methods under analysis operate under the same conditions. We argue that a more formal approach is needed: a *reference model* for the platform considered should be explicitly defined. The reference model should formally define the sensors and the actuators that the control software can access along with the relative value ranges and, possibly, noise models. Ideally, the control software produced by the automatic design methods under analysis should interact with the platform hardware via a common API. This prevents that the experimenter introduces a bias by allowing a method to access resources or information that is not available to other methods or to use them in a more creative and profitable way.

3.2. Precise Definition

An automatic design method should be precisely defined so as to enable the reproducibility of its results. In particular, an automatic design method should be univocally identified by a name, should be clearly defined in all its parts, and should properly pinpoint the reference model(s) to which it can be applied. Ideally, an implementation should be made publicly available. This would guarantee that an automatic design method can be used by researchers other than those who originally proposed it, and can be, therefore, included in objective comparative studies performed by third parties. By studying the literature presented in Section 2, we realized that, excluding very few cases, automatic design methods proposed so far have been tested in a single study by authors who introduced them. In each study, excluding very few exceptions, researchers either present an original design method or develop a variant of a previously presented one. It is extremely rare in the literature that an automatic design method is used "as is" in multiple subsequent studies on different tasks – i.e., without undergoing any *ad hoc*, manual, per-task modification. Moreover, we are not aware of any case in which an automatic design method is included in a study performed by a third party. We find particularly revealing the fact that automatic design methods are not typically given a name by their proponents: it does not make much sense to name a method that is supposed to live within the time span of a single research study. We are quite sure that researchers would name the methods they propose if there were a concrete expectation for them to be included in subsequent studies and applied to several tasks, possibly by a third party. The fact that a method is typically tested on a single task in the original paper in which it is proposed makes it impossible to apprise its qualities as an automatic method. In particular, it makes it impossible to apprise whether the method is an *ad hoc* solution for the single task considered or it is general enough and able to address a class of tasks. Clearly, a method that applies to a single task and needs to be manually re-instantiated and/or fine-tuned to be applied to another task cannot legitimately qualify as an automatic method. To serve the purpose of the research on the automatic design of control software for robot swarm, its experimental practice should take the above considerations into account. In particular, it should prescribe that an automatic design method is tested on multiple tasks without undergoing any *ad hoc*, manual, per-task modification. Moreover, each newly proposed method should be compared with those that have been previously proposed so that a clear picture of the state of the art is always available to the community. Independent evaluations performed by third parties would be extremely valuable.

3.3. Benchmarks

Another important issue that should be addressed by the community concerns the tasks on which automatic design methods should be tested and compared. A convincing and informative experimental analysis should ideally be based on a large set of different tasks. It should eventually allow the experimenter to

make conclusions on the ability of the design methods under analysis to produce control software for a generic task of interest. Obviously, whether a task can be possibly performed by a swarm of robots depends on the capabilities of the robots, as formally characterized by the reference model. In other terms, a reference model implicitly determines the class of tasks that can be performed by a swarm of robots conforming to the reference model itself. An informative comparison should be based on a representative subset of tasks sampled from this class. We are convinced that the definition of publicly available datasets of benchmark tasks would be highly beneficial for the research on the automatic design of control software for robot swarms and would significantly speed up its progress. Each benchmark task should specify its target platform(s) via a reference model. At each moment in time, the research community should have access to an up-to-date record of the performance yielded by each automatic design method that has been tested on each benchmark task. In this respect, having names that univocally identify design methods is paramount. It could be convenient to conceive programs that automatically generate benchmark tasks within a class. For example, consider a program that generates instances of a search and retrieve task by randomly sampling, according to appropriately defined probability distributions: (i) size and shape of the environment, (ii) number and position of the obstacles, (iii) number and the positions of the targets, (iv) initial placement of the robots, and (v) position of the safe area to which retrieved targets should be carried. Such a program could be used to generate multiple sets of tasks, all sharing the same statistical properties. The experimenter could obtain two disjoint sets: one to be used to automatically design control software, and one to be used to test it.

The definition of benchmark tasks to be used by the whole research community could be based on widely used and commercially available robots. Indeed, some platforms have been considered a *de facto* standard by many research labs for years. For example, the Khepera and the e-puck have been adopted in a large share of swarm robotics studies. More recently, the kilobot is becoming popular and the Thymio appears to meet all the requirements to become widely adopted in swarm robotics research. As an alternative, standard benchmark tasks could be implemented via remote controlled labs [e.g., see Zeiger et al. (2009), Kulich et al. (2013), Casini et al. (2014), and Casan et al. (2015)]. For example, imagine a confined area in which a robotic arm can position obstacles and objects as remotely specified by the experimenter. A robot swarm could then act in the area operated by the control software to be tested. The robotic arm would be in charge of placing the robots at desired initial positions for each run and of recharging them when needed.

Clearly, we do not wish to go so far as to state that all the research in swarm robotics should be performed on standard benchmarks and be based on the aforementioned platforms or via remote controlled labs. Indeed, research on previously unexplored tasks and the development of original custom-made robots are essential elements for the advancement of the domain. Yet, we are deeply convinced that routinely testing new proposals on standard benchmarks and common platforms is the only way to consolidate results and to enable an ordered and structured accumulation of knowledge.

3.4. Robot Experiments

A crucial aspect that has been often overlooked in the literature is the ability of a design method to overcome the reality gap, that is, the unavoidable difference between the models used in computer-based simulations and reality. The reality gap is an issue that is particularly relevant when studying off-line automatic design methods. Indeed, one of the major challenges for an off-line design method is to use computer-based simulations to design control software that once installed on the robots will behave as expected. Ideally, the robot swarm should behave in reality as its simulated counterpart. Unfortunately, this challenge is far from being overcome. Potentially, the reality gap is an issue also in the study of on-line methods as a large share of the research work on these methods relies on computer-based simulations. Due to the reality gap, automatic design methods (either on-line or off-line methods) cannot be properly assessed only via simulations, at least in this historical moment. We are firmly convinced that robot experiments should have a much more prominent role in the research on the automatic design of control software for robot swarms. They should be the core of the whole empirical practice in the domain. Increasing the prominence of robot experiment has unfortunately a downside: it raises the cost of research. Yet, we are convinced that, at least in this phase of the development of the domain, robot experiments are indispensable and their cost unavoidable. It is common in many domains that researchers need costly resources to preform their studies. For example, research in particle physics and in observational astronomy require having access to a collider and a telescope, respectively. These resources are expensive – much more than a robot swarm. Most research groups do not own them and need to get access to those shared by other groups. Even those that do own the resources, often are not fully satisfied with them and need to get access to alternative resources owned by other groups. Indeed, each collider and telescope is different (energy range, resolution, location, etc.) and researchers often need to access more than one of them to confirm observations or to calibrate their methods and tools. It is, therefore, customary that these resources are shared in the context of international cooperation programs. Although the domains of particle physics and observational astronomy are deeply different in many respects from swarm robotics, we are convinced that the mechanisms used to share resources within these two domains could serve as a model also for our community. In particular, we envision international cooperation programs within the research community that investigates the automatic design of control software for robot swarms. Research groups could share their resources with partner groups so that each participant in the cooperation program can test their automatic design methods on different platforms and/or under different experimental conditions. When the research groups own identical robots, they could join their resources and perform experiments with a swarm larger than the one they own. The collaboration would be the ideal context to compare the different automatic design methods developed by the partners and conceive possible cross-fertilizations.

4. CONCLUSION

In this article, we reviewed the most notable achievements in the automatic design of control software for robot swarms. These achievements show that automatic methods are a viable and promising approach to the design of control software for robot swarms. Unfortunately, the literature on the automatic design of control software for robot swarms appears to be scattered and composed by isolated contributions: with few exceptions, no comparison between design methods are provided and new ideas and methods are not properly assessed against a well-established state of the art. It is our contention that the lack of an empirical practice hinders the progress of the domain.

In the body of the article, we highlighted four issues that need to be addressed to establish a proper empirical practice for the automatic design of control software for robot swarms: (i) every study that proposes or applies an automatic design method should clearly define a reference model for the robotic platform considered. (ii) Every automatic design method should be precisely defined in all its parts and parameters, and univocally identified by a name. (iii) Libraries of standard benchmarks should be defined and adopted by the community for assessing newly proposed methods and ideas. (iv) Robot experiments should be the ultimate way to assess methods for the automatic design of control software for robot swarms and should be an essential element of any research study in the domain.

We are convinced that a solid, well-established, and consistently applied empirical practice would allow the community to promote the best ideas proposed so far, to focus on promising directions, and to attract further researches and investments to the domain of the automatic design of control software for robot swarms.

AUTHOR CONTRIBUTIONS

The two authors contributed equally to the preparation of the manuscript.

FUNDING

MB acknowledges support from the Belgian F.R.S.–FNRS, of which he is a Senior Research Associate.

REFERENCES

Baldassarre, G., Trianni, V., Bonani, M., Mondada, F., Dorigo, M., and Nolfi, S. (2007). Self-organised coordinated motion in groups of physically connected robots. *IEEE Trans. Syst. Man Cybern. B Cybern.* 37, 224–239. doi:10.1109/TSMCB.2006.881299

Berman, S., Kumar, V., and Nagpal, R. (2011). "Design of control policies for spatially inhomogeneous robot swarms with application to commercial pollination," in *IEEE International Conference on Robotics and Automation – ICRA* (Piscataway, NJ: IEEE), 378–385.

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

Bongard, J. C. (2013). Evolutionary robotics. *Commun. ACM* 56, 74–83. doi:10.1145/2492007.2493883

Brambilla, M., Brutschy, A., Dorigo, M., and Birattari, M. (2015). Property-driven design for swarm robotics: a design method based on prescriptive modeling and model checking. *ACM Trans. Auton. Adapt. Syst.* 9, 17. 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., Montanier, J.-M., Liu, W., and Winfield, A. F. (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

Capi, G. (2007). Multiobjective evolution of neural controllers and task complexity. *IEEE Trans. Robot.* 23, 1225–1234. doi:10.1109/TRO.2007.910773

Casan, G. A., Cervera, E., Moughlbay, A. A., Alemany, J., and Martinet, P. (2015). "ROS-based online robot programming for remote education and training," in *IEEE International Conference on Robotics and Automation – ICRA* (Piscataway, NJ: IEEE), 6101–6106.

Casini, M., Garulli, A., Giannitrapani, A., and Vicino, A. (2014). A remote lab for experiments with a team of mobile robots. *Sensors (Basel)* 14, 16486–16507. doi:10.3390/s140916486

Christensen, A. L., and Dorigo, M. (2006). "Evolving an integrated phototaxis and hole-avoidance behavior for a swarm-bot," in *Artificial Life – ALIFE* (Cambridge, MA: MIT Press), 248–254.

Christensen, A. L., Duarte, M., Costa, V., Rodrigues, T., Gomes, J., Silva, F., et al. (2016). *A Sea of Robots. AAAI Video Competition*. Best Robot Video. Available at: https://youtu.be/JBrkszUnms8

Clark, R. J., Arkin, R. C., and Ram, A. (1992). "Learning momentum: online performance enhancement for reactive systems," in *IEEE International Conference on Robotics and Automation – ICRA* (Piscataway, NJ: IEEE), 111–116.

Di Mario, E., and Martinoli, A. (2014). Distributed particle swarm optimization for limited-time adaptation with real robots. *Robotica* 32, 193–208. doi:10.1017/S026357471300101X

Di Mario, E. L., Navarro, I., and Martinoli, A. (2015). "A distributed noise-resistant particle swarm optimization algorithm for high-dimensional multi-robot learning," in *IEEE International Conference on Robotics and Automation – ICRA* (Piscataway, NJ: IEEE), 5970–5976.

Doncieux, S., and Mouret, J.-B. (2014). Beyond black-box optimization: a review of selective pressures for evolutionary robotics. *Evol. Intell.* 7, 71–93. doi:10.1007/s12065-014-0110-x

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

Duarte, M., Costa, V., Gomes, J. C., 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. (2014a). "Evolution of hierarchical controllers for multirobot systems," in *Artificial Life – ALIFE* (Cambridge, MA: MIT Press), 657–664.

Duarte, M., Oliveira, S. M., and Christensen, A. L. (2014b). "Hybrid control for large swarms of aquatic drones," in *Artificial Life – ALIFE* (Cambridge, MA: MIT Press), 785–792.

Elfwing, S., Uchibe, E., Doya, K., and Christensen, H. I. (2011). Darwinian embodied evolution of the learning ability for survival. *Adapt. Behav.* 19, 101–120. doi:10.1177/1059712310397633

Ferrante, E., Guzmán, E. D., Turgut, A. E., and Wenseleers, T. (2013). "GESwarm: Grammatical evolution for the automatic synthesis of collective behaviors in swarm robotics," in *Genetic and Evolutionary Computation – GECCO* (New York: ACM), 17–24.

Ferrante, E., Turgut, A., Duéñez-Guzmán, E., 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., Brambilla, M., Brutschy, A., Garattoni, L., Miletitch, R., Podevijn, G., et al. (2015). AutoMoDe-chocolate: automatic design of control software for robot swarms. *Swarm Intell.* 9, 125–152. doi:10.1007/s11721-015-0107-9

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

Gauci, M., Chen, J., Li, W., Dodd, T. J., and Groß, R. (2014a). "Clustering objects with robots that do not compute," in *Autonomous Agents and Multiagent Systems – AAMAS* (Richland, SC: IFAAMAS), 421–428.

Gauci, M., Chen, J., Li, W., Dodd, T. J., and Groß, R. (2014b). Self-organized aggregation without computation. *Int. J. Robot. Res.* 33, 1145–1161. doi:10.1177/0278364914525244

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

Haasdijk, E., Bredeche, N., and Eiben, A. (2014). Combining environment-driven adaptation and task-driven optimisation in evolutionary robotics. *PLoS ONE* 9:e98466. doi:10.1371/journal.pone.0098466

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

Hauert, S., Zufferey, J.-C., and Floreano, D. (2008). Evolved swarming without positioning information: an application in aerial communication relay. *Auton. Robots* 26, 21–32. doi:10.1007/s10514-008-9104-9

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 – ANTS, Vol. 7461 of LNCS* (Berlin: Springer), 252–259.

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

König, L., and Mostaghim, S. (2009). Decentralized evolution of robotic behavior using finite state machines. *Int. J. Intell. Comput. Cybern.* 2, 695–723. doi:10.1108/17563780911005845

Koos, S., Cully, A., and Mouret, J.-B. (2013a). Fast damage recovery in robotics with the t-resilience algorithm. *Int. J. Robot. Res.* 32, 1700–1723. doi:10.1177/0278364913499192

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

Kuhn, T. (1962). *The Structure of Scientific Revolutions*. Chicago, IL: University of Chicago Press.

Kulich, M., Chudoba, J., Kosnar, K., Krajnik, T., Faigl, J., and Preucil, L. (2013). Syrotek-distance teaching of mobile robotics. *IEEE Trans. Educ.* 56, 18–23. doi:10.1109/TE.2012.2224867

Lee, J., and Arkin, R. C. (2003). "Adaptive multi-robot behavior via learning momentum," in *IEEE/RSJ International Conference on Intelligent Robots and Systems – IROS* (Piscataway, NJ: IEEE), 2029–2036.

Lehman, J., Risi, S., D'Ambrosio, D., and Stanley, K. O. (2013). Encouraging reactivity to create robust machines. *Adapt. Behav.* 21, 484–500. doi:10.1177/1059712313487390

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

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. (2001). Learning in behavior-based multi-robot systems: policies, models, and other agents. *Cogn. Syst. Res.* 2, 81–93. doi:10.1016/S1389-0417(01)00017-1

Nolfi, S., and Floreano, D. (2000). *Evolutionary Robotics*. Cambridge MA: MIT Press.

Parker, L. E. (1996). "Task-oriented multi-robot learning in behavior-based systems," in *IEEE/RSJ International Conference on Intelligent Robots and Systems – IROS* (Piscataway, NJ: IEEE), 1478–1487.

Parker, L. E. (1997). L-ALLIANCE: task-oriented multi-robot learning in behavior-based systems. *Adv. Robot.* 11, 305–322. doi:10.1163/156855397X00344

Pugh, J., and Martinoli, A. (2009). Distributed scalable multi-robot learning using particle swarm optimization. *Swarm Intell.* 3, 203–222. doi:10.1007/s11721-009-0030-z

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

Reina, A., Valentini, G., Fernàndez-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

Silva, F., Duarte, M., Correia, L., Oliveira, S. M., and Christensen, A. L. (2016). Open issues in evolutionary robotics. *Evol. Comput.* 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

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

Teo, J., and Abbass, H. (2004). Automatic generation of controllers for embodied legged organisms: a pareto evolutionary multi-objective approach. *Evol. Comput.* 12, 355–394. doi:10.1162/1063656041774974

Teo, J., and Abbass, H. (2005). Multiobjectivity and complexity in embodied cognition. *IEEE Trans. Evol. Comput.* 9, 337–360. doi:10.1109/TEVC.2005.846902

Trianni, V. (2008). *Evolutionary Swarm Robotics*. Berlin: Springer.

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

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. (2009). Self-organising sync in a robotic swarm. A dynamical system view. *IEEE Trans. Evol. Comput.* 13, 722–741. doi:10.1109/TEVC.2009.2015577

Usui, Y., and Arita, T. (2003). "Situated and embodied evolution in collective evolutionary robotics," in *International Symposium on Artificial Life and Robotics* (Oita: Dept. of Electrical and Electronic Engineering, Oita University), 212–215.

Waibel, M., Keller, L., and Floreano, D. (2009). Genetic team composition and level of selection in the evolution of cooperation. *IEEE Trans. Evol. Comput.* 13, 648–660. doi:10.1109/TEVC.2008.2011741

Watson, R., Ficici, S., and Pollack, J. (1999). "Embodied evolution: embodying an evolutionary algorithm in a population of robots," in *IEEE Congress on Evolutionary Computation – CEC*, Vol. 1 (Piscataway, NJ: IEEE), 335–342.

Watson, R., Ficici, S., and Pollack, J. (2002). Embodied evolution: distributing an evolutionary algorithm in a population of robots. *Rob. Auton. 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

Wischmann, S., Stamm, K., and Wörgötter, F. (2007). "Embodied evolution and learning: the neglected timing of maturation," in *Advances in Artificial Life – ECAL, Vol. 4648 of LNCS* (Berlin: Springer), 284–293.

Zeiger, F., Schmidt, M., and Schilling, K. (2009). Remote experiments with mobile-robot hardware via internet at limited link capacity. *IEEE Trans. Ind. Electron.* 56, 4798–4805. doi:10.1109/TIE.2009.2027898

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

The reviewer HH declared a past co-authorship with one of the authors MB to the handling Editor, who ensured that the process met the standards of a fair and objective review.