Designing control software for robot swarms

Software engineering for the development of automatic design methods

Darko Bozhinoski
IRIDIA, Université Libre de Bruxelles
Bruxelles, Belgium
darko.bozhinoski@ulb.ac.be

Mauro Birattari
IRIDIA, Université Libre de Bruxelles
Bruxelles, Belgium
mbiro@ulb.ac.be

ABSTRACT

Over the past decade, swarm robotics has emerged as a promising engineering discipline. In this paper, we discuss the current research challenges and state-of-the-art in automatic design methods for swarm control software. This subject has recently received increasing attention from the swarm robotics community. We make a contribution towards the debate by proposing two lines of research from a software engineering point of view.

KEYWORDS

swarm robotics, software engineering, design by optimization

Swarm robotics is a direction in collective robotics that takes inspiration from the self-organized behaviors of social animals. In swarm robotics, a large number of robots are deployed to accomplish a mission that is beyond the capabilities of a single robot [1, 2, 8]. Because a single robot is not able to accomplish the mission on its own, the robots must cooperate. The collective behavior of the swarm is the result of the local interactions that each robot has with its neighboring peers and with the environment. 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. The research in swarm robotics is motivated by the fact that the self-organized and distributed nature of a robot swarm is deemed to yield three important and sought-after properties: reliability, scalability, and flexibility [3]. Due to these properties, swarm robotics is an appealing approach for operating a large number of autonomous robots in environments in which (i) the risk that individual robots fail or are lost is high, (ii) supporting infrastructures are hard or impossible to set up, and (iii) communication has limited bandwidth or short range.

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 [2]. Typically, designers proceed by trial and error: an individual-level behavior is iteratively improved and tested until the desired collective behavior is obtained. This approach is closer to craftsmanship than to engineering: the quality of the result strongly depends on the experience and intuition of the designer. Moreover, this trial-and-error process is time consuming, costly, and lacks repeatability and consistency.

The main reason why robot swarms are challenging to design is their the self-organized and distributed nature. The core issue in swarm robotics is the individual/swarm dichotomy. The relation between individual robots and the swarm that they constitute is a complex one: a swarm is definitely much more than the sum of its parts. The requirements are typically expressed at the swarm level by specifying the mission that the swarm, as a whole, has to perform. However, the swarm is a collective entity and, as such, the swarm itself cannot be pre-programmed, only the individual robots can. The designer's task is therefore indirect: they have to design the individual-level behaviors of the robots that, through a complex set of robot-robot and robot-environment interactions, result in the desired collective behavior of the swarm. Bridging the gap between swarm-level requirements and individual-level behavior is currently the main open issue in swarm robotics. To date, no comprehensive, reliable, and satisfactory solution has been given to this problem [2].

Automatic design is a promising approach for designing control software for robot swarms. In this approach, the design problem is cast into an optimization problem and the solution is obtained using optimization algorithms. 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 of the individual robot. The goal of the optimization algorithm is to find an instance of control software that maximizes an appropriate performance measure. To date, the automatic design of control software for robot swarms has been mostly studied in the framework of evolutionary swarm robotics [9], which is the application of evolutionary robotics [6] to robot swarms. In the classical evolutionary swarm robotics, the control software of each individual robot is a neural network that takes sensor readings as an input and returns actuation commands as an output. The parameters of the neural network are obtained via an evolutionary algorithm that optimizes a mission-specific objective function. The optimization process relies on computer-based simulation. Once simulation shows that the swarm is able to perform the given mission, the neural network is uploaded to the robots and the actual real-world performance of the swarm is assessed. 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. [7] 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. 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 [5] is an approach to evolutionary computation that promotes diversity instead of performance, while multi-objective optimization focuses on problems in which multiple, possibly conflicting objectives are to be optimized. However, the vast majority of these works in which automatic design methods have been introduced are tailored to a specific application scenario, without the possibility to make a generalization of the solution in a particular application domain. That means they were focused on answering scientific questions that do not directly belong in automatic design. For example, an important share of works 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 [10]. These questions are 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.

We consider two lines of research that should be conducted in the software engineering community to help researchers in swarm robotics to conceive and develop effective automatic design methods. These lines of research are based on ideas that have already appeared in the swarm robotics literature and have been delineated in [4].

Defining formal abstractions. Software engineers need to provide tools, methodologies and frameworks for designers to be able to model automatic design methods for swarm control software. We argue that precise formal abstractions are needed to specify the environment and the elements of which the system consists. The range of information defined with these formal abstractions need to include elements like the definition of the sensors and the actuators that the control software can access along with the relative value ranges and, possibly, noise models. In particular, an automatic design method should be defined in all its parts, and should properly pinpoint the swarm model(s) to which it can be applied. Furthermore, there is a need for a definition of a formal language for specifying the requirements and constraints in swarm robotics. The language should provide ways to specify both functional and non-functional requirements. A designer should be able to specify the attributes of the set of robots participating in the mission (e.g. battery level, types of sensors and actuators included, the different models of the sensors/actuators etc.), the goals of the mission that should be executed by the swarm and its constraints. Software engineers should provide tools and methods that are able to transform all these constraints and parameters and synthesize an objective function that should be optimized by an optimization process.

Defining benchmarks for evaluation. The development of a solid, well-established, and consistently applied empirical practice to assess and compare methods is of paramount importance in the field. It is essential for swarm designers to be able to share benchmark problems, datasets, implementations, and results. One of the biggest challenges that should be addressed here is to compare different automatic design methods under the same conditions. It appears obvious that different methods under analysis must be given the same resources: computation time, memory, simulator and simulation models, number and kind of CPUs, operating system, hardware infrastructure, etc. 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. We consider that a future research line should be addressing the following key questions that 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 missions? This means that an automatic design method should be defined in a way to enable the reproducibility of its results. By studying the state of the art [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. The fact that a method is typically tested on a single mission, it makes it impossible to apprise whether the method is an ad hoc solution for the single mission considered or it is general enough and able to address a class of missions without undergoing any ad hoc, manual, per-mission modification. To serve the purpose of the research on the automatic design of control software for robot swarm, its experimental practice should prescribe that an automatic design method is tested on multiple missions without undergoing any ad hoc, manual, per-mission 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 available to the community. A convincing and informative experimental analysis should ideally be based on a large set of different missions which are representative subset of the domain. It should eventually allow the experimenter to make conclusions on the ability of the automatic design methods to produce control software for a generic mission of interest. Defining benchmark missions for the evaluation of different automatic design methods for swarm control software should be the main aspect in developing new software engineering methodologies, principles and tools for swarm missions. Software engineers should develop principles and tools that automatically generate benchmark missions within a class of missions. For example, consider a tool that generates instances of a search and retrieve mission 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 tool could be used to generate multiple sets of missions, 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.

We are firmly convinced that in order to make progress in the field of automatic design methods for swarm control software, software engineering must play a crucial role in the development process. A well-established software engineering discipline can make a contribution to the development of automatic design methods in the following directions: (i) defining formal abstractions that could be reused in different missions and projects and (ii) identifying and defining benchmarks that are important to be considered when designing and evaluating swarm control software. We envision that the results of the proposed research will make a contribution in three directions: (i) development of a classification framework that will allow designers to categorize and reuse methods for their experiments; (ii) isolation of components in automatic design methods that will allow designers to perform cross-fertilization between ideas and will contribute towards a coherent development of methods and (iii) tighter integration between research activities of different research groups.

Acknowledgements. The project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 681872). Mauro Birattari acknowledges support from the Belgian *Fonds de la Recherche Scientifique* – FNRS.

REFERENCES

[1] Gerardo Beni. 2004. From swarm intelligence to swarm robotics. In *International Workshop on Swarm Robotics*. Springer, 1–9.

[2] Manuele Brambilla, Eliseo Ferrante, Mauro Birattari, and Marco Dorigo. 2013. Swarm robotics: a review from the swarm engineering perspective. *Swarm Intelligence* 7, 1 (2013), 1–41.

[3] Tom De Wolf and Tom Holvoet. 2006. Design patterns for decentralised coordination in self-organising emergent systems. In *International Workshop on Engineering Self-Organising Applications*. Springer, 28–49.

[4] Gianpiero Francesca and Mauro Birattari. 2016. Automatic design of robot swarms: achievements and challenges. *Frontiers in Robotics and AI* 3 (2016), 29.

[5] Joel Lehman and Kenneth O Stanley. 2011. Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary computation* 19, 2 (2011), 189–223.

[6] Stefano Nolfi and Dario Floreano. 2000. *Evolutionary robotics: The biology, intelligence, and technology of self-organizing machines.* MIT press.

[7] Matt Quinn, Lincoln Smith, Giles Mayley, and Phil Husbands. 2003. Evolving controllers for a homogeneous system of physical robots: Structured cooperation with minimal sensors. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* 361, 1811 (2003), 2321–2343.

[8] Erol Şahin. 2004. Swarm robotics: From sources of inspiration to domains of application. In *International workshop on swarm robotics*. Springer, 10–20.

[9] Vito Trianni. 2008. *Evolutionary swarm robotics: evolving self-organising behaviours in groups of autonomous robots*. Vol. 108. Springer.

[10] Vito Trianni. 2014. Evolutionary robotics: model or design? *Frontiers in Robotics and AI* 1 (2014), 13.