

Automatic off-line design of robot swarms: exploring the transferability of control software and design methods across different platforms

Miquel Kegeleirs*

IRIDIA, Université libre de Bruxelles
Brussels, Belgium
miquel.kegeleirs@ulb.be

David Garzón Ramos*

IRIDIA, Université libre de Bruxelles
Brussels, Belgium
david.garzon.ramos@ulb.be

Lorenzo Garattoni

Toyota Motor Europe
Brussels, Belgium
lorenzo.garattoni@toyota-europe.com

Gianpiero Francesca

Toyota Motor Europe
Brussels, Belgium
gianpiero.francesca@toyota-europe.com

Mauro Birattari ✉

IRIDIA, Université libre de Bruxelles
Brussels, Belgium
mauro.birattari@ulb.be

Abstract—Automatic off-line design is an attractive approach to implementing robot swarms. In this approach, a designer specifies a mission for the swarm, and an optimization process generates suitable control software for the individual robots through computer-based simulations. Most relevant literature has focused on effectively transferring control software from simulation to physical robots. For the first time, we investigate (i) whether control software generated via automatic design is transferable across robot platforms and (ii) whether the design methods that generate such control software are themselves transferable. We experiment with two ground mobile platforms with equivalent capabilities. Our measure of transferability is based on the performance drop observed when control software and/or design methods are ported from one platform to another. Results indicate that while the control software generated via automatic design is transferable in some cases, better performance can be achieved when a transferable method is directly applied to the new platform.

Index Terms—Automatic design, swarm robotics, transferability, AutoMoDe, evolutionary robotics.

I. INTRODUCTION

A robot swarm [1], [2] is a highly redundant group of robots that operates autonomously without relying on centralized control or external infrastructure. Instead, swarm individuals rely on local sensing and communication to self-organize [3].

*MK and DGR contributed equally to this work and should be recognized as co-first authors. Original software was implemented by MK. `Chocolate` and `EvoStick` were developed by GF. The experiments were performed by MK with the assistance of DGR. The manuscript was drafted by DGR and revised by MK and MB. All authors contributed to the development of the ideas, read the manuscript, and provided comments. The research was directed by MB. Correspondence to mauro.birattari@ulb.be.

The project has received partial funding from the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (DEMIURGE Project, grant agreement No 681872) and from Belgium’s Wallonia-Brussels Federation through the ARC Advanced project GbO-Guaranteed by Optimization. The authors acknowledge support from the Belgian Fonds de la Recherche Scientifique – FNRS and from the Colombian Ministry of Science, Technology and Innovation – Minciencias.

By acting collectively, robots can accomplish tasks that they could not accomplish individually [4].

Designing the collective behavior of a swarm is particularly challenging. No universally applicable methodology exists for developing the control software of the individual robots so that a desired collective behavior emerges [5]. Typically, designers manually refine control software until the desired collective behavior is obtained. This trial-and-error design process is costly, time-consuming, and does not guarantee reproducible or transferable results. Automatic off-line design [6] is an appealing alternative. In this approach, the problem of designing control software for individual robots is re-formulated as an optimization problem. Given mission specifications and a platform description, an optimization algorithm searches for suitable control software for the robots. The automatic design process aims to produce control software that maximizes swarm performance in the mission at hand, according to a mission-specific performance metric provided as part of the mission’s formal specification.

Neuroevolution [7] is the traditional approach to the automatic off-line design of robot swarms. In this approach, control software takes the form of an artificial neural network with parameters tuned via artificial evolution. More recently, *automatic modular design* [8] has been proposed as an alternative to neuroevolution. In modular design, an optimization algorithm selects and tunes predefined software modules into a specific control architecture (e.g., finite-state machines [8] or behavior trees [9]). Both in neuroevolution and the modular approach, the design process is conducted via computer-based simulations. The resulting control software is then transferred to physical robots and assessed in the target environment.

Control software produced via automatic off-line design suffers from the *reality gap* [10]. Unavoidable differences between the design environment (simulation) and the deployment environment (physical robots) can cause a performance drop



Fig. 1. From left to right, e-puck and Mercator robots.

in the swarm [8], [11]–[14]. When comparing design methods, the smaller the performance drop, the greater a method’s ability to cross the reality gap.

Building on this idea, we study the *transferability* of control software and automatic design methods across different robot platforms. We contend that the process of transferring control software across different platforms gives rise to challenges akin to those presented by the reality gap. When discussing transferability, we consider cases where the control software is designed for one platform and then deployed on another, or when an automatic design method conceived for one platform is applied to another. This is reminiscent of the reality gap problem, which emerges when designing control software in simulation for execution on real robots.

In this preliminary study, we first investigate the conditions under which modular and neuroevolutionary design methods conceived for a given robot platform can produce control software for a different one. Then, we investigate whether control software produced via automatic design can be directly transferred from one platform to another. The two platforms considered are ground robots endowed with equivalent sensing and actuation capabilities.

II. EXPERIMENTS

We consider two swarms: one comprising three e-puck [15], [16] robots, and the other, three Mercator [17] robots. We can formally describe the sensing and actuation capabilities of the two platforms with the same reference model RM 1.2 [18]—see Table I—which defines the inputs and outputs on which control software operates [8]. This is key to enabling the transferability between the two platforms. The e-puck and Mercator (Figure 1) differ in size, with the e-puck being roughly one-third the size of the Mercator. They also differ in linear speed and sensor range. However, their speed/size and sensor-range/size ratios are approximately the same.

We design control software for e-pucks and Mercators using two automatic methods originally conceived for the e-puck: *Chocolate* and *EvoStick*. *Chocolate* [11] is a modular design method from the *AutoMoDe* [19] family and *EvoStick* [8] is an implementation of the neuroevolutionary approach. We select these two methods because they have been largely used in the automatic design of collective behaviors

TABLE I
REFERENCE MODEL RM 1.2. INPUT AND OUTPUT VALUES OF E-PUCKS’ (EP) AND MERCATORS’ (ME) CONTROL SOFTWARE.

Input	Value-EP	Value-ME	Description
<i>prox</i>	$((0, 1); (-1, 1) \pi)$	$((0, 1); (-1, 1) \pi)$	proximity vector
<i>light</i>	$((0, 1); (-1, 1) \pi)$	$((0, 1); (-1, 1) \pi)$	light vector
<i>gnd</i>	$\{b, g, w\}$	$\{b, g, w\}$	ground reading
<i>n</i>	$[0, 2]$	$[0, 2]$	no. neighbors
<i>V</i>	$((0, 1); (-1, 1) \pi)$	$((0, 1); (-1, 1) \pi)$	neighbors vector
Output	Value-EP	Value-ME	Description
$v_{k \in \{l, r\}}$	$(-10, 10)$ cm/s	$(-30, 30)$ cm/s	target velocity

Period of the control cycle: 0.1 s.

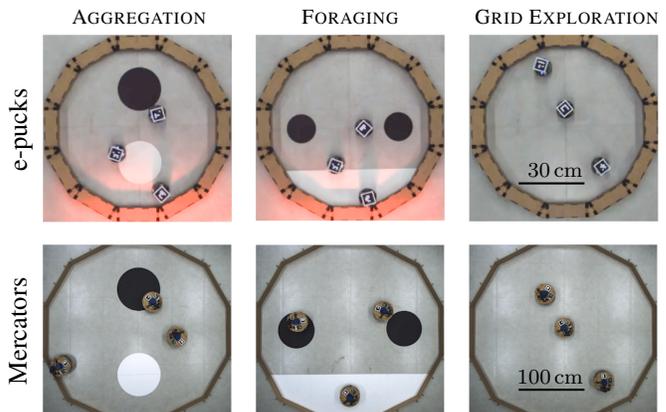


Fig. 2. Experimental scenarios in AGGREGATION, FORAGING, and GRID EXPLORATION. The workspace of the e-pucks is about one-third the size of Mercators’ workspace. The pictures show an example of the initial configuration at the beginning of each mission.

for e-pucks [19]—both in simulation and reality. Previous results confirm that they can generate control software for various missions, including aggregation, foraging, and coverage [8], [11]–[13], [20]. In our experiments, we consider three missions: AGGREGATION, FORAGING, and GRID EXPLORATION—see Figure 2. In AGGREGATION, robots must aggregate on a black spot. In FORAGING, robots must iteratively travel between small black spots and a larger white region. In GRID EXPLORATION, robots must explore the environment and continuously visit every cell of a grid. We adjust the size of the environment according to the relative size of the robots: the e-pucks operate in an environment that is one-third the size of the Mercators’ one. The performance of the two swarms is computed using the same performance measures. We produce a total of 120 instances of control software using *Chocolate* and *EvoStick*—ten for each platform, mission, and method. We assess each instance once in simulation (expected performance) and once with physical robots (real performance). We conduct simulations in *ARGoS3* [21]—a simulator widely used in swarm robotics research.

III. RESULTS AND DISCUSSION

When assessed in simulation, the control software produced by both *Chocolate* and *EvoStick*, for both e-pucks and

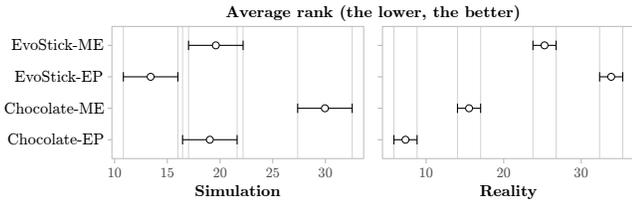


Fig. 3. Comparison of control software produced by `EvoStick` and `Chocolate` for e-pucks (EP) and Mercators (ME). We aggregate the results obtained in the three missions using a Friedman test. For each method and platform, we present average ranks and 95% confidence intervals. The lower, the better.

Mercators, demonstrates meaningful behavior and effectively performs the missions. When assessed in reality, the control software produced by `Chocolate` maintains satisfactory behavior on both platforms. In contrast, control software produced by `EvoStick` does not typically reproduce simulation results on either platform. The differences between simulation and reality are known effects of the reality gap: previous research has shown that design methods from the `AutoMoDe` family are more robust to the reality gap than those based on neuroevolution [12]–[14]. The different degree of robustness to the reality gap between `Chocolate` and `EvoStick` had been only reported for e-pucks. Here, we show that similar results also apply to Mercators. Demonstrative videos are available for download in the Supplementary material¹.

To investigate the extent to which `Chocolate` and `EvoStick` are transferable from the e-puck—the platform for which they were originally conceived—to the Mercator, we compare the performance of the control software they produce for the two platforms. We aggregate the performance across the three missions considered using a Friedman test [22]—see Figure 3. In simulation, the swarm of e-pucks performs better than the one of Mercators, both for `Chocolate` and `EvoStick`. This result was expected, as the two design methods were conceived for the e-puck and were applied to Mercators without any adaptation. In the experiments with the physical robots, the swarm of e-pucks performs better than the one of Mercators when they execute control software produced by `Chocolate`. On the other hand, the swarm of Mercators performs better than the e-pucks when they execute control software produced by `EvoStick`. This result was unexpected and suggests that `EvoStick`, although originally conceived for the e-pucks, is more robust to the reality gap when adopted to design control software for Mercators. This indicates that the effects of the reality gap are not only method-dependent but also platform-dependent. By comparing the relative performance of `Chocolate` and `EvoStick`, we can conclude that, under the experimental conditions considered in the study, `Chocolate` transfers better from e-pucks to Mercators than `EvoStick`—see Figure 3.

We also investigate the extent to which the control software automatically produced by `Chocolate` and `EvoStick` can be transferred between e-pucks and Mercators. To this end,

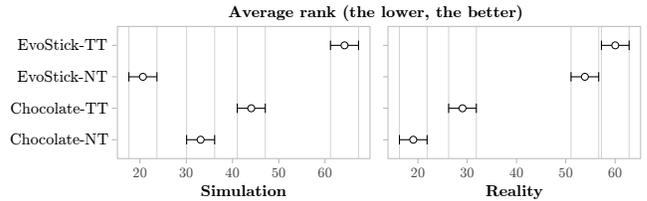


Fig. 4. Comparison of control software produced by `EvoStick` and `Chocolate` when assessed on the platform for which it has been produced (NT, for *not transferred*) and on the counterpart (TT, for *transferred*)—regardless whether the design was performed for e-pucks or Mercators. We aggregate the results obtained in the three missions using a Friedman test. We present average ranks and 95% confidence intervals. The lower, the better.

we transfer the control software produced for e-pucks to Mercators, and *vice versa*. We assess the control software produced by the two methods both on the platform for which it has been produced and on the counterpart. Also in this case, we do this in simulation and reality. We aggregate the performance across the three missions using a Friedman test—see Figure 4. When assessed in simulation, the control software produced by `EvoStick` performs better than the one produced by `Chocolate`. After transferring the control software, we observe that the one produced by `Chocolate` performs better than the one produced by `EvoStick`—see Figure 4. The performance drop caused by transferring the control software is significantly larger for `EvoStick` than for `Chocolate`. We observe a rank inversion between the two design methods. It is known that `EvoStick` can achieve a better performance in simulation by *overfitting* the design process to the simulated model of the e-puck [12], [14]. We argue that this overfitting prevented a proper transfer of the control software to Mercator—as it happens when transferring control software between simulation and reality. In the experiments with the physical robots, `Chocolate` and `EvoStick` show a performance drop after transferring the control software between e-pucks and Mercators. However, the control software produced by `Chocolate` transfers better than the one produced by `EvoStick`—see Figure 4.

These preliminary results indicate that automatic design methods and the control software they produce can be transferred from one robot platform to another—provided that the two have equivalent sensing and actuation capabilities. The best results are obtained by transferring a method, that is, by applying a method originally designed for a platform to another one—as opposed to transferring the control software it produces for the original platform to the other one. We will also investigate whether protocols to predict the robustness of design methods to the reality gap [14] can be used to predict the transferability of control software across platforms.

USE OF AI TECHNOLOGIES IN THE WRITING PROCESS

During the preparation of this work, the authors used OpenAI ChatGPT to proofread and edit for language issues and stylistic inconsistencies—see Supplementary material. After using this tool, the authors reviewed and edited the manuscript as needed and take full responsibility for the content.

¹Supplementary material: <https://iridia.ulb.ac.be/supp/IridiaSupp2023-001>

REFERENCES

- [1] E. Şahin, “Swarm robotics: from sources of inspiration to domains of application,” in *Swarm Robotics: SAB 2004 International Workshop*, ser. Lecture Notes in Computer Science, E. Şahin and W. M. Spears, Eds., vol. 3342. Berlin, Germany: Springer, 2005, pp. 10–20.
- [2] G. Beni, “From swarm intelligence to swarm robotics,” in *Swarm Robotics: SAB 2004 International Workshop*, ser. Lecture Notes in Computer Science, E. Şahin and W. M. Spears, Eds., vol. 3342. Berlin, Germany: Springer, 2005, pp. 1–9.
- [3] M. Dorigo, M. Birattari, and M. Brambilla, “Swarm robotics,” *Scholarpedia*, vol. 9, no. 1, p. 1463, 2014.
- [4] M. Dorigo, G. Theraulaz, and V. Trianni, “Swarm robotics: past, present, and future [point of view],” *Proceedings of the IEEE*, vol. 109, no. 7, pp. 1152–1165, 2021.
- [5] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo, “Swarm robotics: a review from the swarm engineering perspective,” *Swarm Intelligence*, vol. 7, no. 1, pp. 1–41, 2013.
- [6] M. Birattari, A. Ligot, D. Bozhinoski, M. Brambilla, G. Francesca, L. Garattoni, D. Garzón Ramos, K. Hasselmann, M. Kegeleirs, J. Kuckling, F. Pagnozzi, A. Roli, M. Salman, and T. Stützle, “Automatic off-line design of robot swarms: a manifesto,” *Frontiers in Robotics and AI*, vol. 6, p. 59, 2019.
- [7] V. Trianni, *Evolutionary Swarm Robotics*. Berlin, Germany: Springer, 2008.
- [8] G. Francesca, M. Brambilla, A. Brutschy, V. Trianni, and M. Birattari, “AutoMoDe: a novel approach to the automatic design of control software for robot swarms,” *Swarm Intelligence*, vol. 8, no. 2, pp. 89–112, 2014.
- [9] A. Ligot, J. Kuckling, D. Bozhinoski, and M. Birattari, “Automatic modular design of robot swarms using behavior trees as a control architecture,” *PeerJ Computer Science*, vol. 6, p. e314, 2020.
- [10] N. Jakobi, P. Husbands, and I. Harvey, “Noise and the reality gap: the use of simulation in evolutionary robotics,” in *Advances in Artificial Life: Third European Conference on Artificial Life*, ser. Lecture Notes in Artificial Intelligence, F. Morán, A. Moreno, J. J. Merelo, and P. Chacón, Eds., vol. 929. Berlin, Germany: Springer, 1995, pp. 704–720.
- [11] G. Francesca, M. Brambilla, A. Brutschy, L. Garattoni, R. Miletitch, G. Podevijn, A. Reina, T. Soleymani, M. Salvaro, C. Pinciroli, F. Mascia, V. Trianni, and M. Birattari, “AutoMoDe-Chocolate: automatic design of control software for robot swarms,” *Swarm Intelligence*, vol. 9, no. 2–3, pp. 125–152, 2015.
- [12] A. Ligot and M. Birattari, “Simulation-only experiments to mimic the effects of the reality gap in the automatic design of robot swarms,” *Swarm Intelligence*, vol. 14, pp. 1–24, 2020.
- [13] K. Hasselmann, A. Ligot, J. Ruddick, and M. Birattari, “Empirical assessment and comparison of neuro-evolutionary methods for the automatic off-line design of robot swarms,” *Nature Communications*, vol. 12, p. 4345, 2021.
- [14] A. Ligot, “Assessing and forecasting the performance of optimization-based design methods for robot swarms: experimental protocol & pseudo-reality predictors.” Ph.D. dissertation, Université Libre de Bruxelles, Brussels, Belgium, 2023.
- [15] F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz, S. Magnenat, J.-C. Zufferey, D. Floreano, and A. Martinoli, “The e-puck, a robot designed for education in engineering,” in *ROBOTICA 2009: Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions*, P. Gonçalves, P. Torres, and C. Alves, Eds. Castelo Branco, Portugal: Instituto Politécnico de Castelo Branco, 2009, pp. 59–65.
- [16] L. Garattoni, G. Francesca, A. Brutschy, C. Pinciroli, and M. Birattari, “Software infrastructure for e-puck (and TAM),” IRIDIA, Université Libre de Bruxelles, Brussels, Belgium, Tech. Rep. TR/IRIDIA/2015-004, 2015.
- [17] M. Kegeleirs, R. Todesco, D. Garzón Ramos, G. Legarda Herranz, and M. Birattari, “Mercator: hardware and software architecture for experiments in swarm SLAM,” IRIDIA, Université libre de Bruxelles, Brussels, Belgium, Tech. Rep. TR/IRIDIA/2022-012, 2022.
- [18] K. Hasselmann, A. Ligot, G. Francesca, D. Garzón Ramos, M. Salman, J. Kuckling, F. J. Mendiburu, and M. Birattari, “Reference models for AutoMoDe,” IRIDIA, Université Libre de Bruxelles, Brussels, Belgium, Tech. Rep. TR/IRIDIA/2018-002, 2018.
- [19] M. Birattari, A. Ligot, and G. Francesca, “AutoMoDe: a modular approach to the automatic off-line design and fine-tuning of control software for robot swarms,” in *Automated Design of Machine Learning and Search Algorithms*, ser. Natural Computing Series, N. Pillay and R. Qu, Eds. Cham, Switzerland: Springer, 2021, pp. 73–90.
- [20] G. Spaey, M. Kegeleirs, D. Garzón Ramos, and M. Birattari, “Evaluation of alternative exploration schemes in the automatic modular design of robot swarms,” in *Artificial Intelligence and Machine Learning: BNAIC 2019, BENELEARN 2019*, ser. Communications in Computer and Information Science, B. Bogaerts, G. Bontempi, P. Geurts, N. Harley, B. Lebicot, T. Lenaerts, and G. Louppe, Eds. Cham, Switzerland: Springer, 2020, vol. 1196, pp. 18–33.
- [21] C. Pinciroli, V. Trianni, R. O’Grady, G. Pini, A. Brutschy, M. Brambilla, N. Mathews, E. Ferrante, G. A. Di Caro, F. Ducatelle, M. Birattari, L. M. Gambardella, and M. Dorigo, “ARGoS: a modular, parallel, multi-engine simulator for multi-robot systems,” *Swarm Intelligence*, vol. 6, no. 4, pp. 271–295, 2012.
- [22] W. J. Conover, *Practical Nonparametric Statistics*, 3rd ed., ser. Wiley Series in Probability and Statistics. New York, NY, USA: John Wiley & Sons, 1999.