Evolving Swarm Behavior for Simulated Spiderino Robots

Midhat Jdeed, Arthur Pitman, and Wilfried Elmenreich
Institute for Networked and Embedded Systems / Lakeside Labs
Alpen-Adria-Universität Klagenfurt
Klagenfurt, Austria
Email: *firstname.lastname*@aau.at

***Abstract*—In swarm robotics research, simulation is often used to avoid the difficulties of designing swarm behavior in the real world. However, designing the controller of swarm members remains a non-trivial task due to the complex interactions between members and the emerging behavior itself. In this paper, FRamework for EVOlutionary design (FREVO) is used as tool for the design, simulation and optimization of swarm behavior for a given problem. This paper demonstrates how FREVO can be used to develop and optimize the behavior of the entire swarm simultaneously by applying an evolutionary algorithm. A case study consisting of twenty robots given the task of gathering near a light source while keeping a minimum distance from each other based solely on local information from simplistic sensors is presented. Considering the need of a fitness function to guide any evolutionary process which is a problem-specific function, the robots' controller is evolved according to a fitness function depending on two factors: the robots' proximity to the light source and the ability to keep a minimum distance between the robots. We examine in particular how the problem can be modeled and how evolution can be applied to create a suitable controller for the swarm members.**

***Keywords*–Swarm robotics; Spiderino; Evolutionary optimization;.**

I. Introduction

Inspired by the fascinating collective behavior of fish, birds, ants and bees, swarm robotics have gained an increasing interest in the research community. The defining characteristic of these groups is that the emerging swarm behavior is significantly more complex than that of any individual member [1][2].

Research in swarm robotics can be classified into two main groups. The first one is concerned with hardware and considers aspects, such as locomotion, size, communication and cost [3]. Examples for such hardware platforms for swarm research are Kilobot [4], Colias [5], e-Puck [6], Jasmine [7] and the Spiderino platform [8]. The second group deals with swarm design using software simulation. The behavior and the interactions among swarm members remain a complex topic [9] especially in environments where dynamic interactions occur. In particular, it is often difficult to derive the requirements for an individual robot within a swarm from the desired global behavior. This work focuses on modeling and designing swarm behavior using the FREVO [10] software and tackles the challenge of constructing robot behavior using evolutionary principles which try to develop an optimal solution based on a fitness function.

In the evolutionary process, the main challenge is defining an objective (or fitness) function that is designed to reward a desired behaviour of a swarm, which is highly based on each problem individually. The applicability and performance of a fitness function depends on the employed optimizer, thus, there are no universally suitable fitness functions [11]. Nevertheless, many studies in the field of evolutionary optimization have considered generic methods for fitness function design [12][13]. The author in [13] categorized such methods into a three-dimensional fitness space: *Functional vs. behavioural, Global vs. local, and Explicit vs. implicit*. For instance, an explicit function rewards the way in which a certain goal is achieved, while implicit fitness is focused on how much the goal is reached (e.g., a distance).

Furthermore, in the evolutionary process, moving the evolved controller from simulation to real robots is still a big challenge due to differences between the simulation environment and the real world, an issue is known as the reality gap [14]. Within this paper the reality gap is partially addressed by evolving behavior that can be run as code on real robots, and by the definition of a fitness function that can be also evaluated in an experiment with real robots. Further techniques for addressing this problem include intermittently involving real robots in the evolution process [15] and developing accurate models for sensors and actuators [8]. However, a particular assessment of real hardware behavior is outside the scope of this paper.

Designing swarm behaviour by using evolution is mostly an automatic design method that creates an intended swarm behaviour as a result of a bottom up process starting from interactions between very small components. The process gradually modifies potential solutions until a satisfying result is achieved. Such an evolutionary design approach is based on evolutionary computation techniques and can be done either on individual or on a swarm level. In [16], six components are presented for consideration while designing swarm robotics using evolution:

1) The task description: A highly abstract vision of the problem should be created.
2) The simulation setup: The task description must be transformed into an abstracted problem model.
3) The interaction interface: This interface defines the interaction among agents, i.e., the swarm, as well as their interaction with the environment.
4) The evolvable decision unit: The representation of the system or the agent controller, for example an artificial neural network or finite state machine.
5) The search algorithm: In this task, an optimization method will be applied to the results from the above steps like evolutionary algorithms.
6) The objective function: A problem-specific function, often known as a *fitness function*, that guides the search algorithm to a (semi) optimal solution.

Our approach in this paper is to evolve a controller for a swarm consisting of 20 robots using evolutionary design that follow the six mentioned tasks above. The task is to move within a threshold distance of a light source guided only by simplistic sensors. The process is done in a simulated environment using FREVO. Thus, the main contribution of this paper is to demonstrate our approach using FREVO and how it may be applied to solve tasks related to swarm robotics.

The paper is organized as follows: The next section provides a review of related work. Section III outlines the architecture of Spiderino platform, and FREVO tool is introduced in Section IV. Sections V and VI describe the case study, the implementation process and discuss the results. Conclusions are drawn in Section VII, together with an outline of future work.

II. Related Work

Swarm robotics draws inspiration from nature and seeks to offer novel capabilities, such as self-organization, self-learning and self-reassembly [17]. Much research has been conducted to study different behaviours that can be performed using swarm robotics, such as aggregation, flocking, dispersion, foraging, object clustering and sorting, navigation, path formation, deployment, collective transport of objects [18]–[20]. For example, in [21], the authors present a higher multi-robot organism that is capable of autonomous aggregation and disaggregation. Consequently, evolutionary methods gained an increased interest in the research community as it is an approach to deal with design problem in swarm robotics [22]. Nevertheless, the main challenges remain to overcome open issues, such as scaling in complexity, and having a smooth transition from simulation to the real world [23].

There are several software and frameworks supporting evolutionary design in swarm robotics. AutoMoDe [23] is a software for automatic design which generates modular control software in the form of a probabilistic finite state machine. JBotEvolvern is a Java-based versatile open-source platform for education and research-driven experiments in evolutionary robotics [24], which has been used in many studies [25]–[27]. FREVO, a tool for creating and evaluating swarm behaviour, has been used in several studies as an evolution tool including robotics [28] and pattern generation [29]. In [9], a simulated robot soccer game was implemented to evaluate the capabilities of evolutionary algorithms and artificial neural networks. Moreover, a comparison of two different evolvable controller models based on their performance for a simple robotic problem was presented in [30], where a robot has to find a light source using two luminance sensors. The paper compared the relative advantages of Mealy machines, a type of finite state machine, and a fully meshed artificial neural network.

Nevertheless, several works have been succeeded in exploring the use of evolutionary design for generating a robot controller to perform a task. For example, the authors in [31] introduced an evolutionary approach that demonstrates emergent collective in a swarm of simulated Kilobots. The task was to evolve behaviors of phototaxis and clustering. Also, in [32], an embodied evolution method was introduced and it was shown that using evolved controllers can outperform a hand-designed controller in applications like phototaxis from a random location in an environment. Similar works have been carried out to perform phototaxis using evolution processes [33]–[35].

III. Spiderino Platform

This section describes the Spiderino platform in terms of hardware following by the corresponding software framework to develop a controller using simulated evolution:

A. Hardware

Spiderino is a low-cost research robot based on the smaller variant of the Hexbug Spider toy [36]. Figure 1 depicts the Spiderino robot [8], used for the simulation presented in this work. The main aim of designing Spiderino is to be used in swarm research and education. To provide space for sensors, a larger battery, and a Printed Circuit Board (PCB) with Arduino microcontroller, Wi-Fi module, and motor controller, the toy's head was replaced with a 3D-printed adapter. In terms of locomotion, Spiderino has two degrees of freedom: It may turn its head left or right, with a full turn requiring approximately 3 seconds, and it can move forward or backward relative to the direction it is facing by cycling its legs at 0.06 m/s.

Each Spiderino has 6 four-pin interfaces that generically support a variety of sensors. For the experiments in this paper, we assume that each Spiderino is equipped with 6 CNY70 reflective optical sensors [37], positioned at 45°offsets, each consisting of an infrared emitter and a photo-transistor. By turning on and off the infrared LEDs, a Spiderino may distinguish between light sources and obstacles, such as other Spiderinos or walls.

Each robot is equipped with a lithium polymer battery with a capacity of 750 mAh. Spiderino robots consume between 6 mA and 60 mA, with walking being the most expensive operation. Further details of energy consumption are provided in [8].

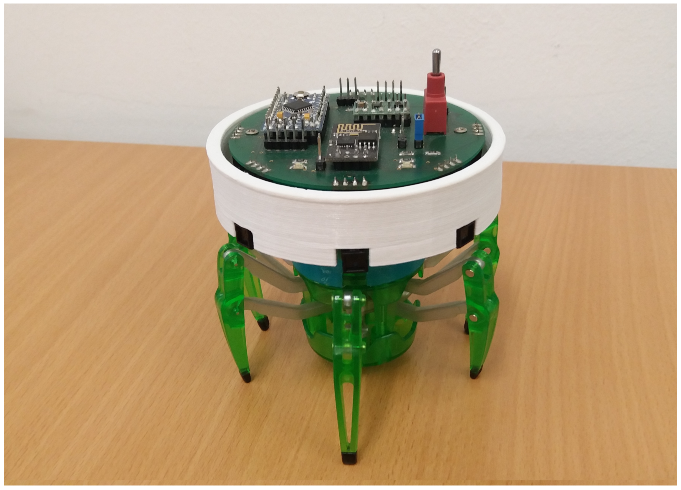


Figure 1. Spiderino robots equipped with CNY70 sensors

B. Software model

We created a software framework for the Spiderino platform to allow the robot's controller to be developed using simulated evolution. Attention was paid to modeling the robot

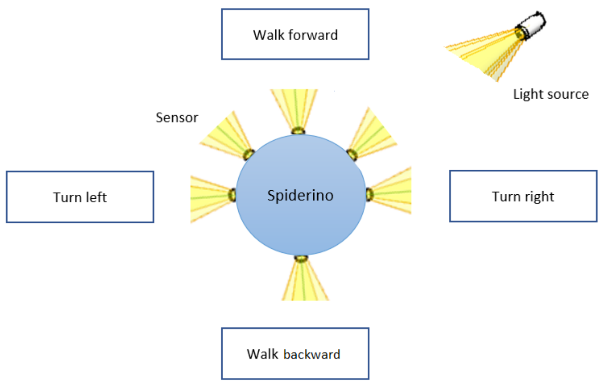


Figure 2. An illustrations model of the sensing phase in Spiderino

accurately to allow validation of the simulation results using actual hardware.

In simulation, each Spiderino is modeled as a solid circular object of radius *spiderinoRadius* (0.06m) moving on a walled flat plane of dimensions *worldWidth* and *worldHeight*. After consideration of the robot's size, weight and locomotion, both friction and momentum were neglected. As outlined in Figure 2, simulation proceeds through *maximumSteps* iterations carried out in two phases:

1) Sense phase: By tracing rays emitted from each of the Spiderino's sensors, the simulator calculates values for both modes of each of the CNY70 sensors.
2) Locomotion phase: Based on the sensor values, a controller may move a Spiderino's head left or right, or walk forwards or backwards. Each movement occurs at speeds given by *spiderinoWalkSpeed* (0.06 m/s) and *spiderinoTurnSpeed* (120°/s) for a period of *stepTime* milliseconds.

To construct a model of the CNY70 sensors, we used the sensor's datasheet [37] as a reference point and gathered empirical evidence about its effectiveness as both a light and proximity sensor. When acting as light sensor, light sources may be detected within a range of few meters depending on factors, such as the light's intensity and ambient light levels (see Figure 3a). When acting as a proximity sensor, the sensor's value is proportional to distance to the closest object (see Figure 3b). Objects may be sensed within a more limited range of approximately 3 cm. If pointed at a light source while measuring proximity, the sensor functions as a light sensor. Objects off-axis up 60° produce amplified readings as described in Figure 3c. For each of the two modes, each sensor value was calculated as the minimum of the values determined for each ray.

IV. FREVO ARCHITECTURE

FREVO is a tool for creating and evaluating systems using evolutionary methods. In order for it to develop a solution, FREVO needs an input consisting of several components as illustrated in Figure 4. First, it is necessary to define the problem where the evaluation context of the agent has to be implemented. Second, a controller representation should be selected that describes the structure of a possible solution. Third, the optimization method must be selected to optimize the chosen controller representation to maximize the fitness returned from the problem definition. Finally, the ranking module is configured to evaluate all agents in a problem and return a ranking of the candidates based on their fitness.

Two types of problem may be modeled in FREVO. The first, a so called *SingleProblem*, where a single candidate controller is evolved, is used in this paper and requires implementations for three functions:

- *evaluateCandidate()* typically utilizes a simulator to evaluate a candidate controller a returns a fitness value that is used to rank the candidate within the population of the generation.
- *replyWithVisualization()* it is called to replay an evaluation with visualization.
- *getMaximumFitness()* specifies the maximum fitness value that can be achieved.

The second type, a *MultiProblem*, evaluates multiple candidates simultaneously, for example, two soccer teams playing against each other as described in [9]. Additional details about using FREVO and constructing a simple simulation are presented in the project's official online tutorial [38].

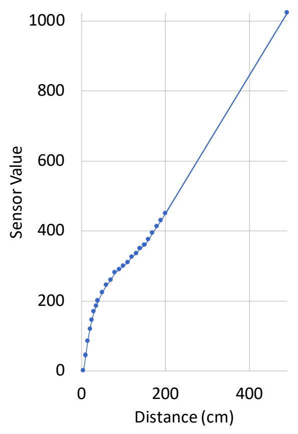
V. CASE STUDY

To perform the case study, a problem component named *SpiderinoSim* has been implemented in FREVO. It is written in Java and models an area where the Spiderinos can move around, a light source, some obstacles and multiple Spiderino robots. The light source can be placed in a specified position, either in the center of a world or at a random location. Spiderinos are placed randomly. *SpiderinoSim* has been modeled as a *SingleProblem* in FREVO, which means that the performance of a Spiderino team is evaluated by an absolute fitness value. In the experiment presented in this paper, twenty Spiderinos, initially placed at random locations, were given the task of keeping a distance of 2 cm from each other and approaching a light source (diameter 0.3 m) in a random location in a walled rectangular world of 3 by 2 meters with obstacles. Each simulation consisted of 1500 steps of 100 milliseconds and, thus, lasted 200 seconds and tested the ability of a candidate controller to guide the Spiderinos to the light source.

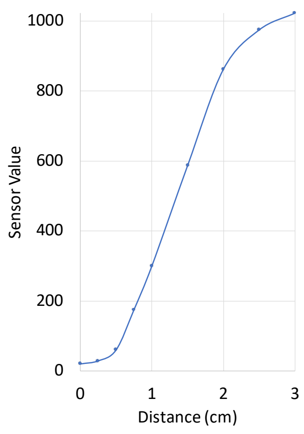
To rank various candidates during the evolution process, FREVO requires a fitness function. The designed fitness function in this case study depends on two components:

- Final distance to the light source: Candidate controllers were rewarded for getting closer to the light.
- Distance between each other: Candidate controllers were rewarded for keeping a distance of 0.02m between each other and penalized for violating such a distance.

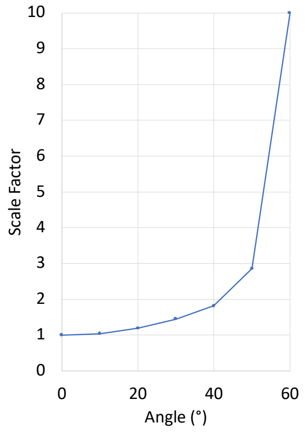
The controller produces two outputs, corresponding to driving one motors for moving forward/backward and turning left or right. A diagram of the Spiderino architecture is given in Figure 5. The inputs of the ANN are twelve values from the six CNY70 sensors, each of them giving a proximity and luminance value.



(a) Light sensitivity



(b) Object sensitivity



(c) Off-axis sensitivity

Figure 3. CNY70 responsiveness

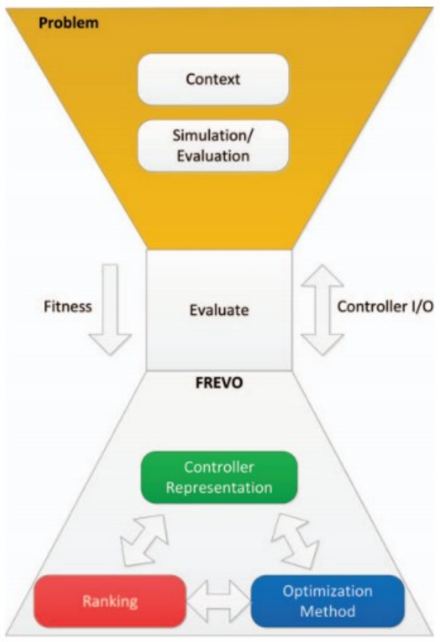


Figure 4. FREVO architecture [10]

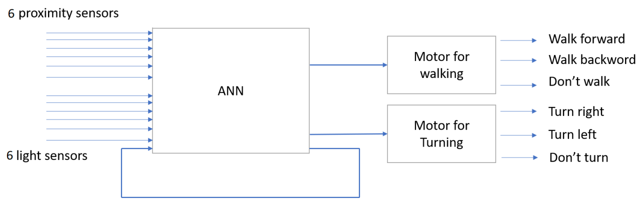


Figure 5. High-level model of a Spiderino in the case study

FREVO offers many evolutionary and representation methods as documented in [10]. ANNs were used as the basis of the controller in all instances. In particular, we utilized FREVO's Fully Meshed Network component with 6 hidden nodes and 2 iterations. Furthermore, in this work we used Cellular Evolutionary Algorithm with two-dimensional Population (CEA2D), a 2-dimensional cellular evolution algorithm, for optimization.

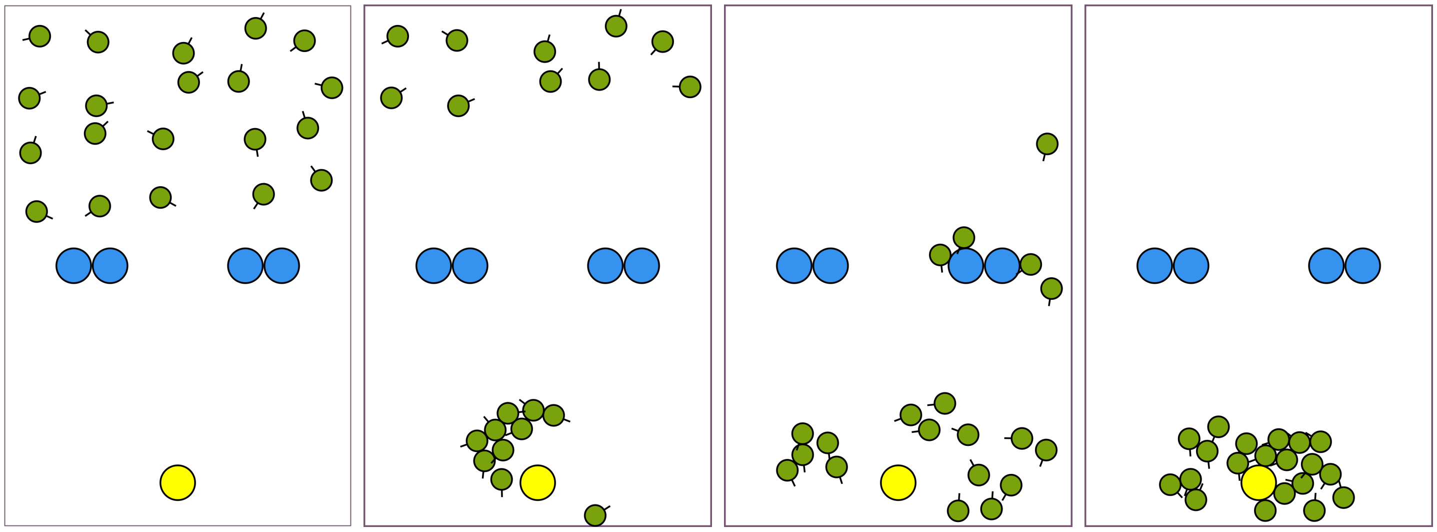
VI. Results and Discussion

Figure 6 illustrates the performance of the Spiderino swarm in after different number of generations. Case 1 shows the initial position, Case 2 shows a partial successfully results where only about half of Spiderinos reach the goal. Case 3 and 4 (Figure 6c and 6d) show a are more successful behavior where eventually all Spiderinos get to the goal. Figure 7 shows the improvement of performance, expressed by the fitness function value over several generations. Evolving 300 generations took about 12 hours on an 8-core machine. While the result, as shown in Figure 6d is satisfactory, there were still small improvements in the achievable fitness after 300+ generations, which is most likely reflecting in small adjustments in robot distribution. Most importantly, the results obtained support that notion that it is possible to evolve a controller for the task using evolution.

VII. Conclusion

In this paper, we have described a method for designing swarm behavior using evolution. In the case study, a swarm of twenty robots evolved the skills required to gather at a light source while keepeing a distance from each other. More specifically, the neural network learnt to interpret its twelve sensory inputs to control its motors. While the task introduced in this paper is rather simplistic it is analogous to the homing task in swarm robotics. Moreover, the evolutionary approach may be applied to a wide variety of problems that may be expressed through a fitness function. In a simulated environment, we evolved controllers for 20 robots to approach a light source by utilizing a fitness function depending on two factors: distance from the light source as well as the distance between each other.

In future work, we hope to cross the reality gap and apply the evolved controllers to real hardware. To extend the case study, we intend to compare the performance of



(a) Case 1: Before Running The Evolution, Initial Positions (b) Case 2: After 118 Generations, 1500 Steps, Fitness 57.7 (c) Case 3: After 307 Generations, 750 Steps (d) Case 4: After 307 Generations, 1500 Steps, Fitness 69.5

Figure 6. Sample final simulation states

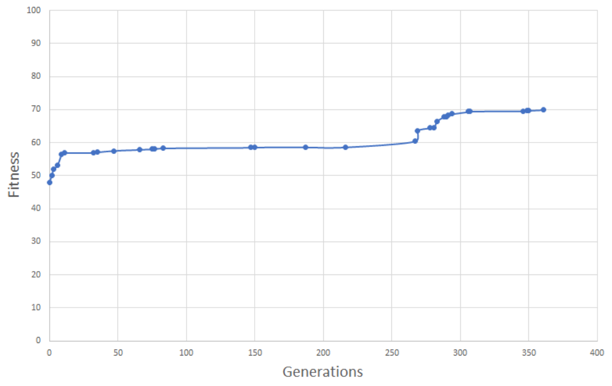


Figure 7. Fitness function values over 361 generation

such controllers with hand-written algorithms and consider factors, such as the time required to reach the light, as well as key swarm properties, such as flexibility and robustness. Another potential work is to rank various candidates during the evolution process using different fitness functions, such as, implementing a multiple-objective function that also takes the energy consumption resulting from locomotion into account.

Acknowledgments

We would like to thank our colleagues Elnaz Khatmi, Markus Müller, Sabrina Huber, and the anonymous reviewers for their constructive comments on an earlier version of this paper. This work was partially supported by Lakeside Labs via the Smart Microgrid Lab and by the Self-Organizing Systems cluster of University of Klagenfurt. The research leading to these results has received funding from the European Union Horizon 2020 research and innovation program under grant agreement No 731946 (Project CPSwarm).

References

[1] Y. Tan, "A survey on swarm robotics," Handbook of Research on Design, Control, and Modeling of Swarm Robotics, vol. 1, 2015, pp. 1–41, DOI: 10.4018/978-1-4666-9572-6.ch001.

[2] H. Hamann, Swarm Robotics: A Formal Approach. Springer, 2018, DOI: 10.1007/978-3-319-74528-2_5.

[3] M. Patil, T. Abukhalil, S. Patel, and T. Sobh, "Hardware architecture review of swarm robotics system: Self reconfigurability, self reassembly and self replication," in Innovations and Advances in Computing, Informatics, Systems Sciences, Networking and Engineering. Springer, 2015, pp. 433–444, DOI: 10.1007/978-3-319-06773-5_58.

[4] M. Rubenstein, C. Ahler, and R. Nagpal, "Kilobot: A low cost scalable robot system for collective behaviors," in Robotics and Automation (ICRA), 2012 IEEE International Conference on. IEEE, 2012, pp. 3293–3298, DOI: 10.1109/ICRA.2012.6224638.

[5] F. Arvin, J. Murray, C. Zhang, and S. Yue, "Colias: An autonomous micro robot for swarm robotic applications," International Journal of Advanced Robotic Systems, vol. 11, no. 7, 2014, p. 113, DOI: 10.5772/58730.

[6] 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 Proceedings of the 9th conference on autonomous robot systems and competitions, vol. 1, no. LIS-CONF-2009-004. IPCB: Instituto Politécnico de Castelo Branco, 2009, pp. 59–65, DOI: 10.7717/peerj-cs.82/fig-5.

[7] S. Kernbach, R. Thenius, O. Kernbach, and T. Schmickl, "Re-embodiment of honeybee aggregation behavior in an artificial micro-robotic system," Adaptive Behavior, vol. 17, no. 3, 2009, pp. 237–259, DOI: 10.1177/1059712309104966.

[8] M. Jdeed, S. Zhevzhyk, F. Steinkellner, and W. Elmenreich, "Spiderino-a low-cost robot for swarm research and educational purposes," in Inteligent Solutions in Embedded Systems (WISES), 2017 13th Workshop on. IEEE, 2017, pp. 35–39, DOI: 10.1109/wises.2017.7986929.

[9] I. Fehérvári and W. Elmenreich, "Evolving neural network controllers for a team of self-organizing robots," Journal of Robotics, vol. 2010, 2010, pp. 1–10, DOI: 10.1155/2010/841286.

[10] A. Sobe, I. Fehérvári, and W. Elmenreich, "FREVO: A tool for evolving and evaluating self-organizing systems," in Self-Adaptive and Self-Organizing Systems Workshops (SASOW), 2012 IEEE Sixth International Conference on. IEEE, 2012, pp. 105–110, DOI: 10.1109/sasow.2012.27.

[11] A. J. Lockett, "Insights from adversarial fitness functions," in Proceedings of the 2015 ACM Conference on Foundations of Genetic Algorithms XIII. ACM, 2015, pp. 25–39, DOI: 10.1145/2725494.2725501.

[12] D. Floreano and J. Urzelai, "Evolutionary robots with on-line self-organization and behavioral fitness," Neural Networks, vol. 13, no. 4-5, 2000, pp. 431–443, DOI: 10.1016/s0893-6080(00)00032-0.

[13] I. Fehérvári, "On evolving self-organizing technical systems," Ph.D. dissertation, Alpen-Adria-Universität Klagenfurt, Nov. 2013.

[14] J.-B. Mouret and K. Chatzilygeroudis, "20 years of reality gap: a few thoughts about simulators in evolutionary robotics," in Proceedings of the Genetic and Evolutionary Computation Conference Companion. ACM, 2017, pp. 1121–1124, DOI: 10.1145/3067695.3082052.

[15] S. Koos, J.-B. Mouret, and S. Doncieux, "The transferability approach: Crossing the reality gap in evolutionary robotics," IEEE Transactions on Evolutionary Computation, vol. 17, no. 1, 2013, pp. 122–145, DOI: 10.1109/tevc.2012.2185849.

[16] I. Fehérvári and W. Elmenreich, "Evolution as a tool to design self-organizing systems," in International Workshop on Self-Organizing Systems. Springer, 2014, pp. 139–144, DOI: 10.1007/978-3-642-54140-7_12.

[17] M. Patil, T. Abukhalil, S. Patel, and T. Sobh, "Ub robot swarmdesign, implementation, and power management," in Control and Automation (ICCA), 2016 12th IEEE International Conference on. IEEE, 2016, pp. 577–582, DOI: 10.1109/icca.2016.7505339.

[18] I. Navarro and F. Matía, "An introduction to swarm robotics," Isrn robotics, vol. 2013, 2012, pp. 1–10, DOI: 10.5402/2013/608164.

[19] L. Bayındır, "A review of swarm robotics tasks," Neurocomputing, vol. 172, 2016, pp. 292–321, DOI: 10.1016/j.neucom.2015.05.116.

[20] Y. Tan and Z.-y. Zheng, "Research advance in swarm robotics," Defence Technology, vol. 9, no. 1, 2013, pp. 18–39, DOI: 10.1016/j.dt.2013.03.001.

[21] S. Kornienko, O. Kornienko, A. Nagarathinam, and P. Levi, "From real robot swarm to evolutionary multi-robot organism," in Evolutionary Computation, 2007. CEC 2007. IEEE Congress on. IEEE, 2007, pp. 1483–1490, DOI: 10.1109/cec.2007.4424647.

[22] V. Trianni, "Evolutionary robotics for self-organising behaviours," Evolutionary Swarm Robotics: Evolving Self-Organising Behaviours in Groups of Autonomous Robots, 2008, pp. 47–59, DOI: 10.1007/978-3-540-77612-3_4.

[23] 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, 2014, pp. 89–112, DOI: 10.1007/s11721-014-0092-4.

[24] M. Duarte, F. Silva, T. Rodrigues, S. M. Oliveira, and A. L. Christensen, "Jbotevolver: A versatile simulation platform for evolutionary robotics," in Proceedings of the 14th International Conference on the Synthesis & Simulation of Living Systems. MIT Press, Cambridge, MA, 2014, pp. 210–211, DOI: 10.7551/978-0-262-32621-6-ch035.

[25] M. Duarte, A. L. Christensen, and S. Oliveira, "Towards artificial evolution of complex behaviors observed in insect colonies," in Portuguese Conference on Artificial Intelligence. Springer, 2011, pp. 153–167, DOI: 10.1007/978-3-642-24769-9_12.

[26] J. Gomes, P. Urbano, and A. L. Christensen, "Evolution of swarm robotics systems with novelty search," Swarm Intelligence, vol. 7, no. 2-3, 2013, pp. 115–144, DOI: 10.1007/s11721-013-0081-z.

[27] T. Rodrigues, M. Duarte, S. Oliveira, and A. L. Christensen, "What you choose to see is what you get: an experiment with learnt sensory modulation in a robotic foraging task," in European Conference on the Applications of Evolutionary Computation. Springer, 2014, pp. 789–801, DOI: 10.1007/978-3-662-45523-4_64.

[28] W. Elmenreich, R. D'Souza, C. Bettstetter, and H. de Meer, "A survey of models and design methods for self-organizing networked systems," in Proceedings of the Fourth International Workshop on Self-Organizing Systems, vol. LNCS 5918. Springer Verlag, 2009, pp. 37–49, DOI: 10.1007/978-3-642-10865-5_4.

[29] W. Elmenreich and I. Fehérvári, "Evolving self-organizing cellular automata based on neural network genotypes," in International Workshop on Self-Organizing Systems. Springer, 2011, pp. 16–25, DOI: 10.1007/978-3-642-19167-1_2.

[30] A. Pintér-Bartha, A. Sobe, and W. Elmenreich, "Towards the light-comparing evolved neural network controllers and finite state machine controllers," in Intelligent Solutions in Embedded Systems (WISES), 2012 Proceedings of the Tenth Workshop on. IEEE, 2012, pp. 83–87, Electronic ISBN: 978-3-902463-09-8.

[31] J. Holland, J. Griffith, and C. O'Riordan, "Evolving collective behaviours in simulated kilobots," in Proceedings of the 33rd Annual ACM Symposium on Applied Computing. Association for Computing Machinery, 2018, pp. 824–831, DOI: 10.1145/3167132.3167223.

[32] R. A. Watson, S. G. Ficici, and J. B. Pollack, "Embodied evolution: Distributing an evolutionary algorithm in a population of robots," Robotics and Autonomous Systems, vol. 39, no. 1, 2002, pp. 1–18, DOI: 10.1016/s0921-8890(02)00170-7.

[33] F. Silva, P. Urbano, L. Correia, and A. L. Christensen, "odneat: An algorithm for decentralised online evolution of robotic controllers," Evolutionary Computation, vol. 23, no. 3, 2015, pp. 421–449, DOI: 10.1162/evco_a_001417.

[34] F. Silva, L. Correia, and A. L. Christensen, "Leveraging online racing and population cloning in evolutionary multirobot systems," in European Conference on the Applications of Evolutionary Computation. Springer, 2016, pp. 165–180, DOI: 10.1007/978-3-319-31153-1_12.

[35] E. Di Paolo, "Spike-timing dependent plasticity for evolved robots," Adaptive Behavior, vol. 10, no. 3-4, 2002, pp. 243–263, DOI: 10.1177/1059712302010003006.

[36] HEXBUG Micro Robotic Creatures, accessed: 26.3.2019. [Online]. Available: https://www.hexbug.com

[37] CNY70 Datasheet, accessed: 26.3.2019. [Online]. Available: https://datasheet.octopart.com/CNY70-Vishay-datasheet-5434663.pdf

[38] FREVO Tutorial, accessed: 26.3.2019. [Online]. Available: https://sourceforge.net/p/frevo/wiki/Tutorials/