Assessment of Evolutionary Processes

Experiments on Self-Organizing Behavior of E-pucks

Hossein Ghaffari Nik
ECE Department of the Volgenau School of Engineering
George Mason University
Fairfax, Virginia, United States of America
hghaffar@masonlive.gmu.edu

Dr. Nathalia Peixoto
ECE Department of the Volgenau School of Engineering
George Mason University
Fairfax, Virginia, United States of America
npeixoto@gmu.edu

***Abstract*—Evolutionary process has become a popular design method for experimenting and automatically synthesizing intelligent controllers for autonomous robots. Such controllers are automatically created using different evolutionary methods without direct programming or in-depth human knowledge of the design. Multi-agent systems and collective behaviors based on swarm intelligence observed in nature are generally ideal candidates for automatic controller design using evolutionary processes. Although the evolutionary process can provide great insight into possible solutions and is a reasonable tool for such experiments, it may not be the most efficient and ideal design tool for every experiment. In this paper, we setup experiments on self-organization of a multi-agent system and evolve three different controllers. We then compare the design effort and results of the three evolved controllers with a traditionally designed controller that performs the same task. We show that the traditionally designed controller outperforms the best case evolutionary-based controller by 10% when measured in overall median fitness level. We also show that the traditionally designed controller is the most reliable as it never violates the design rules. Many studies have been performed in comparison of different evolutionary techniques, but to the best of our knowledge none of them focus on the study of design effort and suitability of such approaches. The evolutionary process is a reasonable design tool only for problems that are too difficult or complicated to be addressed using traditional design methods.**

Keywords—Evolutionary Robotics; Swarm Robotics; Genetic Algorithm; Evolutionary Process; Control Design

I. Introduction

Evolutionary process has become a popular design method for experimenting and automatically synthesizing intelligent controllers for autonomous robots. Such controllers are automatically created using different evolutionary methods without direct programming or in-depth human knowledge of the design. Evolutionary processes are widely used in evolutionary robotics (ER), a research field that bridges artificial life and autonomous robotics. ER can be used to create both the control structure and the control parameters for autonomous robots through the means of an evolutionary process. An evolutionary process generally consists of software simulations on a population-based artificial environment to synthesize controllers for autonomous robots. The robots are then evolved over generations to correctly execute tasks and exhibit desired behaviors with some level of intelligence.

This work was supported in part by the Neural Engineering Laboratory at George Mason University under advisement of Dr. Nathalia Peixoto.

Multi-agent systems and collective behaviors based on swarm intelligence observed in nature are generally ideal candidates for automatic controller design using evolutionary processes [1]–[3]. Swarm robotics (SR) is the research field that deals with such systems. In SR, the multi-agent systems are generally comprised of mostly simple physical robots with a specific desired collective behavior. The collective behavior emerges from a set of elementary rules that govern the interactions of individual robots with each other and their respective environment. A well-constructed evolutionary process can produce decentralized systems with desired emergent properties, such as robustness to individual failures, flexibility and adaptability to environmental changes, or scalability to different group sizes.

Although the evolutionary process can provide great insight into possible solutions and is a reasonable tool for such experiments, it may not be the most efficient and ideal design tool for every experiment. The evolutionary process uses a bottom-up approach where the design is mainly focused on piecing together fundamental systems of actuators and sensors to give rise to a complex and intelligent system. Evolutionary process needs large number of fast, accurate and computationally intensive simulations. Depending on the setup of the evolutionary process the resulting controller may be simple or complex. The synthesized controllers are generally evolved in isolation and are subject to local optimizations where they may not yield the exact desired system. These limitations lead to a black-box solution that is difficult to modify and reverse engineer with a possible reality gap that would prevent applying the controller to a real-world robot and environment.

In contrast to ER, the traditional control design uses a top-down approach. In this approach the designer starts with the big picture (desired collective behavior) and breaks down the system to different subsystems (individual robots and their behaviors). Furthermore, each subsystem is broken down into smaller parts (actuators and sensors) where physical and mathematical models govern the controller, rules of interactions and the collective behavior. This approach requires a prior knowledge of the solution and the breakdown is generally difficult to perform due to the indirect relationships between the rules executed by the robots. However, the traditional control design is still widely used in most swarm robotic designs as it generally outperforms the evolutionary process results.

Currently, the top-down approach is widely used in most swarm robotic designs where individual level behaviors are designed and tested using a trial-and-error process. The automatic design approach of ER as an alternative has been adopted in many cases [4]–[7]. The complexity of classical design and difficulty of transitioning ER from simulations to real world (reality gap) has given rise to a newly proposed automatic modular design (AutoMoDe) approach [8], [9]. AutoMoDe generates an individual-level behavior in the form of a probabilistic finite state machine (PFSM) by searching for the best combination of preexisting parametric modules and uses an optimization algorithm to select the topology of PFSM, the modules to be included, and the value of their parameters. The set of modules and the rules to compose a probabilistic finite state machine represent the bias injected in the automatic design process [5], [6]. Experiments with the vanilla variation of AutoMoDe has shown that while it can perform better than ER approach, it still cannot outperform the classical approach.

In this paper, we setup experiments on self- organization of a multi-agent system and evolve three different controllers similar to [4], [7]. We devised three experiments that evolved controllers for the e-puck educational robot by École Polytechnique Fédérale de Lausanne (EPFL), Switzerland using MATLAB® (MathWorks – Natick, Massachusetts) and V-REP (Virtual Experimentation Platform from Coppelia Robotics – Switzerland). We also developed a traditionally designed controller that performed the same task as the evolved controllers. We then compare the design effort and results of the three evolved controller with a traditionally designed controller. Although the extent of our experiments may be limited, we feel they are sufficient to show that traditionally designed controller outperforms the controllers synthesized using evolutionary process. The evolutionary process and ER are an extremely valuable tool and research field, when it is applied to problems that are challenging for a traditional control design and is not an ideal solution for all ordinary problems.

In the next section, we provide detail information on the setup of the evolutionary process and the controller designed using a traditional design method. In section III we present the results obtained from the simulations with a discussion and comparison of different controllers. Lastly, in section IV we provide a summary and conclusion to our experiment.

II. Methods

Evolutionary process as a design tool can synthesize desired controllers for autonomous robots without direct human programming. This approach differs from the traditional control design were the designer creates the controller using physical and mathematical models and by dividing the whole system into subsystems. Successful results have been reported using ER [4], [7], [10]–[12] and in particular V. Trianni and S. Nolfi have published case studies on self-organizing swarms of s-bots using their proposed design process approach in [4], [7]. This paper reports results of experiments and simulations performed on a similar robotic platform the e-puck. In this section we first introduce the robotic platform that was used in the experiments and the desired collective behavior. We then provide details on the setup of three evolutionary experiments and end with details of the traditionally designed controller.

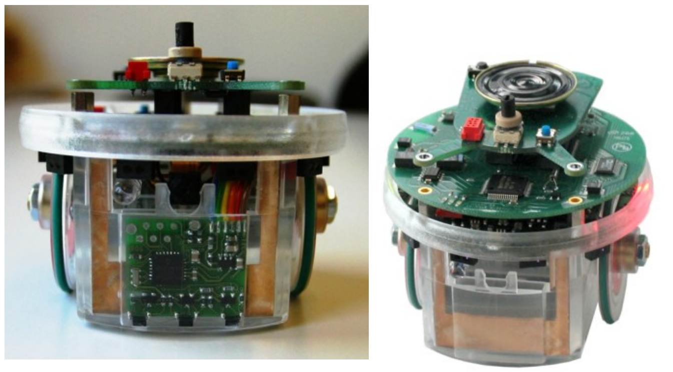
Fig. 1. E-puck the educational robot by École Polytechnique Fédérale de Lausanne (EPFL), Switzerland [3] with a host of sensors and actuators.

A. The Robotic Platform and Desired Collective Behavior

The e-puck as depicted in Fig. 1 is an educational robotic platform used in SR and ER [6], [13]. The e-puck robot has a diameter of 75 mm and is a differential drive robot with a host of sensors and actuators that includes: two stepper motors, loud speaker, three microphones to capture sound, red and green light emitting diodes (LED), color CMOS camera for vision, eight infrared (IR) proximity sensors around the body of the robot for obstacle avoidance and three ground sensors for measuring light intensity and ground color.

The experiments we performed in this paper were setup to devise a decentralized controller for a group of e-pucks to develop a self-organizing synchronization by employing an individual periodic behavior consisting of oscillations along the y axis of the rectangular arena shown in Fig. 2. The arena is a 6×3 meter painted floor surrounded by walls. The ground is painted white for $|y| < 0.2$ m, linearly changes to black until $|y| = 1$ m and is painted black for $|y| > 1$ m. The three ground sensors in the front of e-puck are expected to be used for adjustments to the heading of the robot in producing individual periodic oscillatory movement along the y-axis of the arena. The light intensity measured by these sensors can be used as a positioning system providing y-location and heading. The loud speaker and microphones are one suitable selection for means of communication for synchronizing self-organizing oscillation. The eight IR proximity sensors around the body of the robot are an obvious choice for obstacle avoidance.

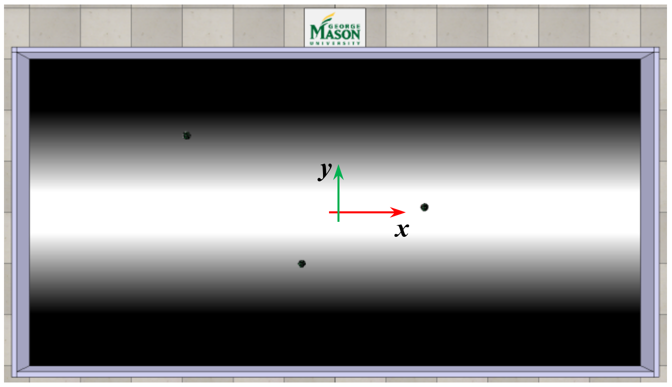

Fig. 2. Top-view snapshot of V-REP simulation illustrating the environment in which the evolutionary experiments were performed. The three dark circles in the arena are 3 individual e-pucks.

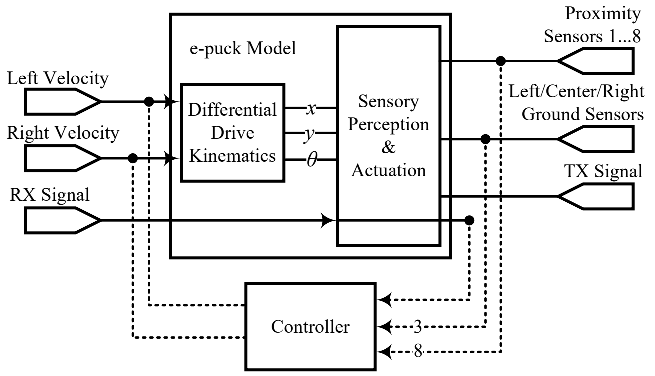

Fig. 3. Simplified block diagram of the e-puck model used in the experiments. The controller at the bottom of the figure is the subject of these experiments and is to be designed.

We setup two evolutionary experiments using MATLAB simulations with an in-house model of the e-puck without sensory noise or actuator disturbances. A simplified input/output block diagram of the e-puck and the desired controller is shown in Fig. 3. The kinematics of the e-puck was modeled as a differential drive robot where left and right angular velocities of the wheels acted as its inputs as described in (1):

$$\begin{aligned} x[n+1] &= x[n] + v \cdot \cos(\omega) \cdot \Delta t \\ y[n+1] &= y[n] + v \cdot \sin(\omega) \cdot \Delta t \\ \theta[n+1] &= \theta[n] + \omega \cdot \Delta t \\ v &= \frac{r}{2}(v_R + v_L) \\ \omega &= \frac{r}{L}(v_R - v_L) \end{aligned} \tag{1}$$

Where v_R and v_L are the angular velocities of the right and left motors respectively, r is the radius of the wheels and L is the separation between the two wheels.

The third evolutionary experiment we setup used an identical model as shown in Fig. 3 but the simulations were carried in V-REP virtual environment. Our traditionally designed controller consisted of a proportional controller and was tested using the same simulation environment.

B. *Evolutionary Experiments and Setup*

We followed the guidance presented in [4] for the setup of the evolutionary experiments performed here. Based on the characteristics of the arena and the desired collective behavior, we selected the right and left motors and the three ground sensors as the sensory-motor system for achieving the individual periodic oscillatory behavior. The left and right motor velocities are controlled based on the perception of the e-puck from the arena using the ground sensors. The ground sensors can decode the grey level into a range of [0, 1], where 0 corresponds to black and white is 1. The maximum angular velocity of the wheels is set to 2π rad/sec. If the obstacle avoidance is also a desired individual behavior, the eight proximity IR sensors can be used for collision avoidance and maneuvering. Finally, for the collective behavior and synchrony among the e-pucks the loud speaker and microphone is used as a communication channel. The communication among the e-pucks is configured as a global binary communication system, where if any or all the e-pucks are signaling they all perceive a 1 as the received signal and a 0 if no one is signaling.

Neural network controllers are an ideal choice for ER [1], [14]. There are two parts to a neural network controller: the genetics or genotype that is evolved in the experiment and potentially holds the desired solution to the problem, and the organism or phenotype that in this case is the robotic system. Evolution is heavily dependent on the distinction between genotype and phenotype, and their relation, that is the genotype-to-phenotype mapping [7]. In these experiments we used a homogeneous set of e-pucks where in each trial all individuals were identical and clones of the same genotype. We selected direct mapping and used a fully connected, feed-forward neural network control structure also called a perceptron network for genotype-to-phenotype mapping. This perceptron directly transforms the sensory inputs into actuator outputs, without recurrent connections or internal states or layers. The sensory neurons are passed through after being multiplied by their corresponding weights, and the output neurons are sigmoid units whose activation is computed by (2):

$$O_j = \sigma\left(\sum_i w_{ij} \cdot I_j + B_j\right), \qquad \sigma(z) \overset{\text{def}}{=} \frac{1}{1+e^{-z}} \tag{2}$$

Where I_i is the activation of the i^{th} input unit, B_j is the bias term, O_j is the activation of the j^{th} output unit, w_{ij} is the weight of the connection between input neuron i and output neuron j, and $\sigma(z)$ is the sigmoid function. There are a total of twelve inputs and three outputs for the neural network controller. Three sensory neurons, I_1 to I_3, are mapped into the three ground sensors. Eight sensory neurons, I_4 to I_{11}, are mapped into the proximity RI sensors and the last sensory neuron, I_{12}, is a binary input and mapped into the microphone of the e-puck for perception of sound. All sensor readings are scaled to the range of [0, 1]. O_1 and O_2 are the activation states of the first two motor neurons corresponding to left and right e-puck motors and are mapped to the range of $\pm v_{max}$ (2π rad/sec). The third motor neuron controls the speaker in such a way that a sound signal is emitted whenever the activation state O_3 is greater than 0.5. Fig. 4 illustrates the input/output chain of the perceptron network that controls the first motor neuron O_1. The same structure is repeated with different weights and biases for the rest of the output motor neurons O_2 and O_3. The connection weights w_{ij} and bias terms B_j are the genotypes and are the genetically encoded parameters. Each parameter is represented with an 8-bit binary code mapped onto a real number ranging in [−10, +10].

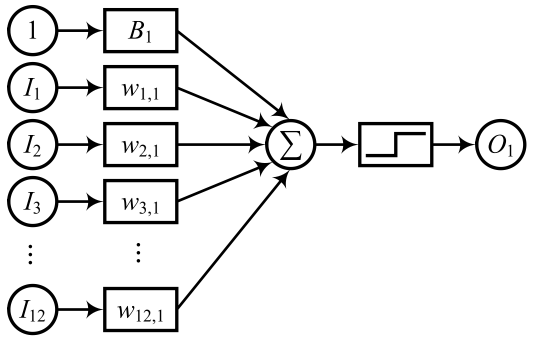

Fig. 4. Input/output chain of the fully connected feed-forward perceptron network that controls the first output motor neuron O_1. The same structure is repeated with different weights and biases for the rest of the output motor neurons O_2 and O_3.

To evaluate the performance of evolved controllers a two component fitness function is used for these experiments that is set as metrics to reward individual motion (F_M) and group synchrony (F_S). The overall fitness of the group is evaluated as the average of these two:

$$F = \frac{1}{2} \cdot F_M + \frac{1}{2} \cdot F_S, \quad F \in [0,1] \tag{3}$$

The movement component F_M relates to the individual motion and rewards e-pucks that move along the y-axis of the arena at the maximum speed:

$$F_M = \frac{1}{R}\sum_{r=1}^{R}\left(\frac{1}{T}\sum_{t=1}^{T}\left(\frac{|\Delta y(t,r)|}{\Delta Y}\right)\right) \tag{4}$$

Where R is the number of robots in the swarm, $\Delta y(t, r)$ is the variation of the y-position of e-puck r at cycle t, and ΔY is the maximum possible variation, which corresponds to the e-puck moving at maximum speed in a direction parallel to the y-axis. This fitness component rewards fast motion along the y-axis. Oscillations during the whole trial are necessary to maximize F_M.

The group synchrony component F_S evaluates the degree of synchronized movement between two e-pucks and is calculated for all possible pairs in the group. The overall F_S of the swarm is taken as the least synchronized pair in the group. The function defined in (5) encodes the movements of an individual e-puck in a way that incorporates the direction of movement (away from or toward the x-axis) and the actual physical y-position of the e-puck. This is necessary to stress the desired synchronized oscillatory movement.

$$d(t,r) = y(t,r) \cdot \frac{\Delta y(t,r)}{\Delta Y} \tag{5}$$

The cross-correlation coefficient ϕ_{r1r2} of two sequences $d(t,r_1)$ and $d(t,r_2)$ is defined as:

$$\phi_{r1r2} = \frac{\Phi_{r1r2}}{\sqrt{\Phi_{r1r1} \cdot \Phi_{r2r2}}},$$
$$\Phi_{r1r2} = \frac{1}{T}\sum_{t=1}^{T}\left(d(t,r_1) \cdot d(t,r_2)\right) \tag{6}$$

The coefficient ϕ_{r1r2} can take values in $[-1, 1]$, where a value of 1 indicates perfect synchrony and a value of -1 indicates perfect asynchrony. F_S is computed using (6) considering (5) and is bounded in [0, 1].

$$F_S = \max\left\{0, \min_{r1 \neq r2} \phi_{r1r2}\right\} \tag{7}$$

F_M and F_S reward movement of the e-pucks and their synchrony from the observer's perspective, without explicitly indicating how to perform a periodic behavior and are both behavioral, external and implicit. In addition to F_M and F_S, we used two other indirect selective pressures needed to stress the desired self-organizing oscillatory behavior. The experiment is given performance fitness of $F = 0$ if any of the e-pucks move over to the black painted area. This is to assert the need to staying only in the white and gradient painted area. If collision avoidance is desired in the evolutionary experiment, the trial also receives fitness of $F = 0$ if there is a collision between e-pucks or the e-pucks and the walls.

To avoid complexity and computationally extensive simulations, the maximum number of e-pucks in each experiment was set to two. In the case where collision avoidance was not used, the arena was not walled and the e-pucks were free to move in infinite x-direction.

C. *Traditionally Designed Controller*

In an effort to compare the ER process and results we developed a traditionally designed proportional controller to mimic one of the evolved behaviors in the evolutionary experiments. There are many sophisticated controllers and complex solutions for creating a decentralized self-organizing oscillatory swarm behavior, but the solution used in this part is simple and similar to an evolved solution.

The controller takes the sensory data of the e-puck at every step and acts on them as the states of the system. For simplicity, a single delay (memory) is devised on the center ground sensor data which enables the controller to know the direction of movement (i.e. moving black-to-white or white-to-black). The proportional controller acts on the error between the left and right ground sensors to make adjustments on the heading of the e-puck and effectively controlling the direction of the movement along the y-axis only. A left-turn is triggered once a threshold is reached on the center ground sensor (< 0.1) to ensure that the e-puck does not cross to the black area and the direction of movement is switched to black-to-white. The same threshold is used in producing a binary signal which is globally perceived by the other e-pucks. This signaling behavior (receiving a binary one on the receiver end) will cause the same left-turn trigger if the e-puck is moving white-to-black. These individual behaviors force the swarm system to the desired self-organized synchronized oscillatory movement.

The collision avoidance mode of the e-puck controller is triggered if an obstacle is detected within the radius of 25 cm from the e-puck. For obstacle avoidance the four front proximity sensors PS_2 through PS_5 are used in conjunction with Braitenberg weights.

$$\begin{bmatrix} \acute{v}_R \\ \acute{v}_L \end{bmatrix} = \begin{bmatrix} 1 & 2 & -2 & -1 \\ -1 & -2 & 2 & 1 \end{bmatrix} \times \begin{bmatrix} PS_2 \\ PS_3 \\ PS_4 \\ PS_5 \end{bmatrix} \tag{8}$$

III. Results

We performed three evolutionary experiments, two of which were done using an in-house MATLAB simulation environment and one using V-REP virtual experimentation platform. We setup the first experiment using without evolving the collision avoidance behavior where the arena was not walled and the e-pucks were free to move in infinite x-direction. We enabled the evolution of collision avoidance for the other two experiments.

We ran each evolutionary experiment through 500 generations. We set the population count of each generation to 100 genotypes each corresponding to a unique controller. The first generation of each experiment was initialized with a randomly selected population of controllers. We evaluated the performance of each controller in every generation through ten instances of trials with randomly selected initial conditions of the e-pucks (position and heading in the arena). We then calculated the fitness of each controller as the average of the ten

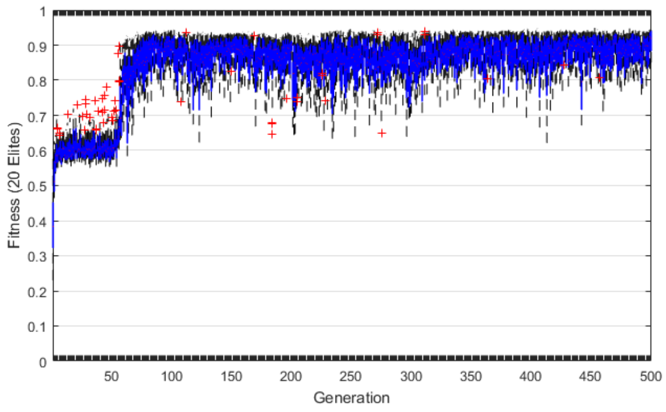

Fig. 5. Simulation results of MATLAB experiment without collision avoidance. Boxplot of the fitness from the 20 top performing controllers in each generation. Outliers are denoted by the plus sign.

trials using (3). Once the fitness of each genotype in the population has been evaluated, we produced the next generation's new population by a combination of selection with elitism and mutation. At each generation, the four best controllers of the population, the elite, are retained in the next generation. The 96 remainder of the new population is generated by the 20 individuals of the previous generation that scored the highest fitness. Each selected genotype reproduces at most five times by applying mutation with 3% probability of flipping a bit. The simulations were performed with a time step of 50 ms for a total duration of 90 seconds without any noise or disturbances introduced.

A. *MATLAB Simulation Results without Collision Avoidance*

We performed the experiment without collision avoidance with the assumption that the e-pucks are in an infinite arena. They were free to move in x-direction without any constrains (no walls) and that the probability of e-pucks colliding with each other was negligible and not crucial to evolution of self-organizing synchronous collective behavior. This assumption enabled much less computationally intensive simulations compared to simulations where collision avoidance was part of the evolutionary experiment. The only indirect selective pressure for this experiment was when the e-pucks moved over to the black painted part and a fitness of $F = 0$ was assigned to the trial. Fig. 5 shows the results from the experiment without collision avoidance.

B. *MATLAB Simulation Results with Collision Avoidance*

In this experiment, we setup the process to evolve the collision avoidance of the controller. We imposed both of the selective pressures on the simulations. If any of the e-pucks moved to the black painted area or if any of the e-pucks collided with the wall or the others the trial would receive a fitness of $F = 0$. These two in conjunction with the fitness function (3) would lead the experiment to evolving self-organizing synchronous collective behavior with collision avoidance. As the search area for weights was increased due to the addition of the proximity sensors, the simulations for this experiment required extremely high computational resources and extended run time. Fig. 6 shows the results from the experiment with collision avoidance.

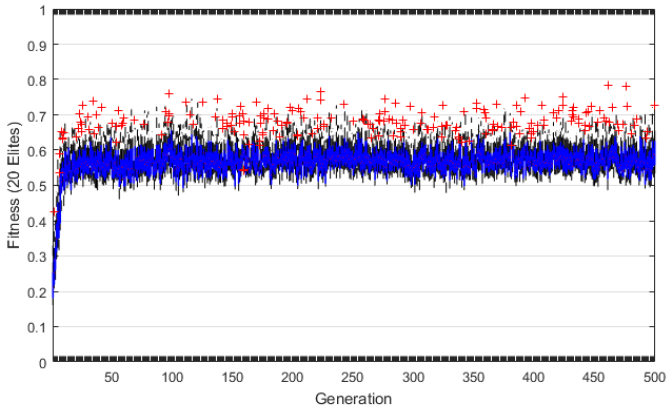

Fig. 6. Simulation results of MATLAB experiment with collision avoidance. Boxplot of the fitness from the 20 top performing controllers in each generation. Outliers are denoted by the plus sign.

C. *V-REP Simulation Results with Collision Avoidance*

We modified and used an e-puck model included with V-REP provided by École Polytechnique Fédérale de Lausanne (EPFL), Switzerland. This model was identical to the one we used in section III.A-B simulations. We used the default physics engine (ODE 2.78) for dynamic calculations and simulation of real-world physics and object interactions. Fig. 7 shows the results from the V-REP experiment with collision avoidance.

D. *Performance Comparison of Evolved and Traditionally Designed Controllers*

We selected the best performing evolved controller of last generation from each evolutionary experiment in sections II.A-C to compare with the traditionally designed controller. We simulated all four controllers 100 times with random initial poses and measured their movement (F_M), synchrony (F_S) and overall fitness ($F_{Overall}$). Fig. 8 is a boxplot summary of the results and compares different fitness levels of each controller. Traditionally designed controller with collision avoidance is denoted by "T", evolved controllers using MATLAB without and with collision avoidance are denoted by "E1" and "E2" respectively and the controller evolved using V-REP with collision avoidance is denoted by "E3".

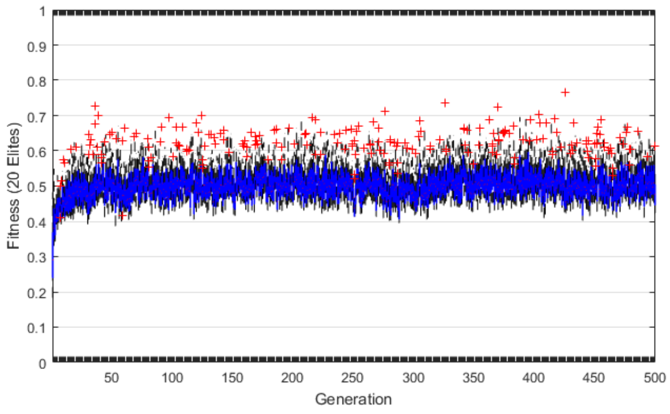

Fig. 7. Simulation results of V-REP experiment with collision avoidance. Boxplot of the fitness from the 20 top performing controllers in each generation. Outliers are denoted by the plus sign.

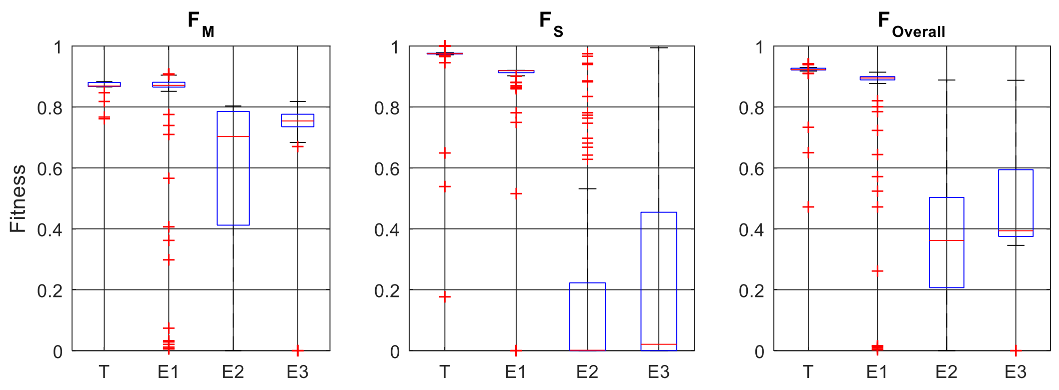

Fig. 8. Performance comparison of evolved and traditionally designed controllers. Traditionally designed controller with collision avoidance is denoted by "T", evolved controllers using MATLAB without and with collision avoidance are denoted by "E1" and "E2" respectively and the controller evolved using V-REP with collision avoidance is denoted by "E3".

E. Discussion

We focused on overall performance, reliability, design effort and adaptability of the controllers. The performance of the controllers was indicated by the fitness function. The reliability was measured based on instances of failures in following the rules set forth. The design effort was measured by the time required to design or evolve the final controller. Finally the adaptability of the controllers based on ease of accommodating possible changes in the robotic system and the environment was a factor in selecting the best performing design method for the given problem.

Traditionally designed controller outperformed all other three evolved controllers both in terms of fitness level and reliability. All three evolved controllers violated the design rules and received fitness of zero at least in one instance. Evolved controller without collision avoidance (E1) is the closest match to the traditionally designed controller. The evolved controllers (E2 and E3) both underperformed due to lack of convergence in the evolutionary process, caused by added complexity of collision avoidance requirement. Although the performance of T and E1 are close to each other, we used a two sample t-test and with a p-value of 0.0019 showed that the two groups are significantly different from each other. Traditionally designed controller (T) outperformed the best evolved controller without collision avoidance (E1) by 10% in median of the overall fitness, without receiving a fitness of zero and smallest variance.

TABLE I. STATISTICS OF OVERALL FITNESS FOR CONTROLELRS

	T	E1	E2	E3
Median	0.9223	0.8949	0.3616	0.3936
Variance	0.0031	0.0447	0.0589	0.0372
Maximum	0.9412	0.9140	0.8883	0.8874
Minimum	0.4718	0	0	0
Outliers	9	14	0	0

The evolutionary experiment without collision avoidance (E1) outperformed the other two evolutionary experiments with collision avoidance. As the evolutionary experiment is a tool for optimizing coefficients in a search area related to weights of the neural network as shown in Fig. 4, larger search area introduces complexity. As the experiment without collision avoidance has a smaller search area, we observed a convergence to the desired solution after 60 generations. However, the other two with collision avoidance did not converge well enough even after 500 generations. One factor in poor performance of the later experiments is the effect of disturbances. In situations where e-pucks get close to walls or each other the obstacle avoidance intervenes and disturbs the established synchrony among the group. Another factor is also related to evolution and the use of signaling capabilities of the e-pucks. From Fig. 8 we can observe that the movement component (F_M) of E2 and E3 are relatively well evolved, but the synchrony component (F_S) is poorly evolved and hence the overall fitness of these two are underperforming compared to the others. We attribute better performance of E3 over E2 to the simulation environment provided by V-REP and the real-world physical engines.

The evolutionary experiments required extremely long and computationally intensive simulations in comparison to the design effort for the traditionally designed controller. The experiment without collision avoidance (E1) completed in 24 hours and faster compared to the other two experiments. As we introduced collision avoidance in E2, the completion of experiment required much longer simulation time (96 hours). The simulation in V-REP with parallel processing and simultaneous simulations required even longer simulation time (over 10 days) due to the rich and comprehensive physical engine used by V-REP.

IV. Conclusion

Evolutionary process as a design tool has gained popularity in evolutionary and swarm robotics for synthesizing intelligent controllers for autonomous robots. Such controllers are automatically created using different evolutionary methods without direct programming or in-depth human knowledge of the design. Although the evolutionary process is a powerful design tool, it may not be the most efficient and ideal solution for every experiment. The evolutionary process uses a bottom-up approach where the design is mainly focused on piecing together fundamental systems of actuators and sensors to give rise to a complex and intelligent system. Evolutionary process needs large number of fast, accurate and computationally intensive simulations. Depending on the setup of the evolutionary process the resulting controller may be simple or complex. The synthesized controllers are generally evolved in isolation and are subject to local optimizations where they may not yield the exact desired system. These limitations lead to a black-box solution that is difficult to modify and reverse engineer with a possible reality gap that would prevent applying the controller to a real-world robot and environment.

In this paper, we setup three experiments on self-organization of a multi-agent system and evolve three different controllers similar to [4], [7]. We selected two simulation environments (MATLAB and V-REP) with different levels of complexities. The goal of this paper was to examine the performance of controllers when compared with a traditionally designed controller. We showed that an evolved controller can provide desired outcome, but it was outperformed by the traditionally designed controller. Higher level of complexity in the desired task required more intensive and longer simulation time. In the case of our self-organizing experiments and outcome, the amount of effort, time and computational resources required for evolutionary experiments was far more than the effort for a traditionally designed controller. Based on our experimental results we demonstrated that the evolutionary process is a reasonable design tool, however it is best used for problems that are too difficult or complicated to be solved by traditional design methods.

REFERENCES

[1] K. Salama and A. M. Abdelbar, “A Novel Ant Colony Algorithm for Building Neural Network Topologies,” in *Swarm Intelligence*, M. Dorigo, M. Birattari, S. Garnier, H. Hamann, M. M. de Oca, C. Solnon, and T. Stützle, Eds. Springer International Publishing, 2014, pp. 1–12.

[2] A. L. Christensen, R. O’Grady, and M. Dorigo, “From Fireflies to Fault-Tolerant Swarms of Robots,” *IEEE Trans. Evol. Comput.*, vol. 13, no. 4, pp. 754–766, Aug. 2009.

[3] L. Sabattini, C. Secchi, M. Cocetti, A. Levratti, and C. Fantuzzi, “Implementation of Coordinated Complex Dynamic Behaviors in Multirobot Systems,” *IEEE Trans. Robot.*, vol. 31, no. 4, pp. 1018–1032, Aug. 2015.

[4] V. Trianni and S. Nolfi, “Engineering the Evolution of Self-Organizing Behaviors in Swarm Robotics: A Case Study,” *Artif. Life*, vol. 17, no. 3, pp. 183–202, Jul. 2011.

[5] J. A. Fernandez-Leon, G. G. Acosta, and M. A. Mayosky, “Behavioral control through evolutionary neurocontrollers for autonomous mobile robot navigation,” *Robot. Auton. Syst.*, vol. 57, no. 4, pp. 411–419, Apr. 2009.

[6] F. Montes-Gonzalez and F. Aldana-Franco, “The Evolution of Signal Communication for the e-puck Robot,” in *Advances in Artificial Intelligence*, I. Batyrshin and G. Sidorov, Eds. Springer Berlin Heidelberg, 2011, pp. 466–477.

[7] V. Trianni and S. Nolfi, “Self-Organizing Sync in a Robotic Swarm: A Dynamical System View,” *IEEE Trans. Evol. Comput.*, vol. 13, no. 4, pp. 722–741, Aug. 2009.

[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 Intell.*, vol. 8, no. 2, pp. 89–112, Mar. 2014.

[9] G. Francesca *et al.*, “AutoMoDe-Chocolate: automatic design of control software for robot swarms,” *Swarm Intell.*, vol. 9, no. 2–3, pp. 125–152, Jun. 2015.

[10] V. Trianni, S. Nolfi, and M. Dorigo, “Evolution, Self-organization and Swarm Robotics,” in *Swarm Intelligence*, C. Blum and D. Merkle, Eds. Springer Berlin Heidelberg, 2008, pp. 163–191.

[11] M. Dorigo *et al.*, “Swarmanoid: A Novel Concept for the Study of Heterogeneous Robotic Swarms,” *IEEE Robot. Autom. Mag.*, vol. 20, no. 4, pp. 60–71, Dec. 2013.

[12] G. Francesca *et al.*, “An Experiment in Automatic Design of Robot Swarms,” in *Swarm Intelligence*, M. Dorigo, M. Birattari, S. Garnier, H. Hamann, M. M. de Oca, C. Solnon, and T. Stützle, Eds. Springer International Publishing, 2014, pp. 25–37.

[13] P. Tarquino and K. Nickels, “Programming an E-Puck Robot to Create Maps of Virtual and Physical Environments,” in *Robot Intelligence Technology and Applications 2*, J.-H. Kim, E. T. Matson, H. Myung, P. Xu, and F. Karray, Eds. Springer International Publishing, 2014, pp. 13–28.

[14] J. Velagic, N. Osmic, and B. Lacevic, “Design of Neural Network Mobile Robot Motion Controller,” in *New Trends in Technologies*, B. Ramov, Ed. InTech, 2010.