Radboud Repository

Radboud University Nijmegen

PDF hosted at the Radboud Repository of the Radboud University Nijmegen

The following full text is a publisher's version.

For additional information about this publication click this link.
http://hdl.handle.net/2066/54833

Please be advised that this information was generated on 2021-04-16 and may be subject to change.

Interleaving Simulated and Physical Environments Improves Evolution of Robot Control Structures

Twan Goosen [a] Rik van den Brule [a] Joris Janssen [a] Pim Haselager [a,b]

[a] *Department of Artificial Intelligence, Radboud University Nijmegen*
[b] *Nijmegen Institute for Cognition and Information*

Abstract

Control structures for physical robots can be evolved in simulated and physical environments. In this study, the interleaving of simulated and physical environments during the course of the evolution of a control structure was examined. This method was compared to the method of 'fine tuning' in a physical environment a control structure that has evolved in complete simulation. Interleaving physical and simulated environments improves performance of the eventual control structure. Possibly, this method allows for the evolved control structure to incorporate and retain advantageous behavioral patterns from both environments.

1 Introduction

Evolutionary robotics is a field of science in which evolutionary methods are exploited in order to synthesize control structures for robots that operate in either a simulated environment or in the physical world. This approach is based on genetic algorithms, originally developed by John Holland [4].

In the present study, a control structure for a physical robot was evolved. A number of epochs throughout the evolutionary process took place in simulation, while generations in between these epochs were evolved in a physical robot in a real environment. The issue under investigation is whether this method results in better performing control structures in comparison to the now more common method of evolving all generations in simulation entirely except for a small number of final generations.

1.1 Evolving control structures in simulated and physical environments

For reasons of efficiency control structures for physical robots are often partly evolved in simulation.

The following method is often applied (see for example [7]): A control structure is evolved in simulation for a number of generations. Subsequently, the evolutionary process is continued in a physical environment for a smaller number of generations (typically about a tenth of the number of preceding generations). The advantage of such a method follows from the relatively small amount of time and work required for evolving in simulation when compared to real world evolution. Miglino, Lund and Nolfi [7] propose a number of techniques to optimize the results of the method. Unfortunately, these techniques are rather complicated and require quite some effort to improve the validity of the simulation. Other authors [5] introduce the term 'reality gap', referring to the discrepancies between simulation and reality that cause evolved control structures that perform well in simulation to fail in reality. Several approaches to this phenomenon can be taken. Jacobi, Husbands and Harvey [5] and Miglino et al. [7] suggest applying noise. Eggenberger and Gómez [2] take a developmental approach.

Another way of combining simulation and physical robots when evolving control structure is by *interleaving* physical and simulated environments over the course of evolution. While assigning most of the work to the simulation, like the method mentioned above, an interleaving approach will *prepare* the eventual control system for the real world from the early stages on.

One notable example of the use of such an 'interleaving' strategy can be found in the work of [12]. In their experiment, the evolutionary process is split into separate phases, each of which handles a specific aspect of evolution. Their approach yields acceptable results, and is faster than if real robots had been used throughout [11]. However, it strongly deviates from Holland's well proven GA method. In addition, human interference is required at every loop of the process, which may prove cumbersome.

1.2 Interleaving simulated and physical environments in evolution

Despite the problems found in the experiments of [12], the principle of interleaving simulated and real robots and environments may have a number of intrinsic pros when compared to, for example, the work of [7]. In the experiments of the latter, great efforts are made to model the physical robot and its environment as accurately as possible. In addition, a carefully selected type of noise must be added to the simulation. While these measures may actually optimize the results while minimizing the number of real world trials that have to be run, they also demand for a certain amount of time, effort and expertise that may not always be available.

We propose a method that, like the method of [12] involves interleaving simulation and real world evolution during the course of evolution, but also allows for comparison with the method of [7]. The main difference with the method of [12] is that except for the nature of the environment and the robot (namely whether these are virtual or physical), all relevant factors are kept equal. Most importantly, the software that controls the robot and contains the control structure and the genetic algorithm and its parameters are equal. No efforts are made to optimally identify the simulation with the real world.

The main object of study in is the effect of interleaving on the fitness of eventual control structures. Over the course of evolution, control structures become increasingly reliant on the robot's environment and morphology. By breaking the chain of growing dependent on the simulation and by preparing, if you will, the control structure for the real world in bite-size chunks, a better performing robot is expected to result. One question that has to be answered first of all, is whether switching back and forth early on in the process improves eventual performance at all.

2 Experimental setup

2.1 The robot and the environment

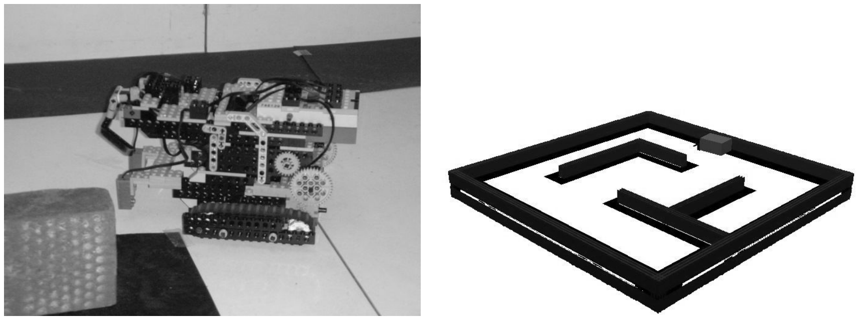
The physical robots used in this experiment are composed of the parts found in the Lego Mindstorms robot kit. Its main component is the programmable *brick*, which provides motor outputs and sensory inputs, and in which the control structure is stored. A great advantage of Lego Mindstorms for the purposes of this study is the availability of a customizable simulation platform, the use of which is discussed in Section 2.1.4. The descriptions below concern the physical robot and environment. The simulated robot and environment are modeled to their physical counterparts.

2.1.1 The environment and task

The robot was placed inside an arena in which it can freely move. This arena was designed in such a way that it allows the robot to behave according to its intended task, which is defined through the fitness function of the genetic algorithm (described in more detail in Section 2.2). The task requires the robot to drive around the arena as far as possible whilst avoiding bumping against the walls that surround it. These walls are about the same height of the robot. Walls are placed inside the arena as well. The robot is unable to cross any of these walls. It is possible however to drive around the walls *inside* the arena, since they do not form enclosed regions. The floor of the arena is white, except for the regions surrounding the walls, which are covered with strips of black paper. The white of the floor and the black of the paper constitute a difference in luminance large enough to be picked up by the light sensors of the robot. Light conditions were kept equal during the entire experiment. The control structure can configured such that bumping against walls can be avoided by retracting upon the perception of black areas.

2.1.2 The robot

Figure 1(a) shows the morphology of the physical robot. Two motor units are connected to the Mindstorms brick, which control the two caterpillar tracks that are positioned at the sides of the vehicle. At the front side



(a) The physical robot positioned in the arena (b) The simulated environment

Figure 1: The physical environment and the simulation

of the robot, three sensors are placed: two light sensors pointing to the floor and one touch sensor, which is activated when pressure is applied to a bumper in front of the robot.

The program that was downloaded to the programmable brick contains the control structure for the robot, which processes incoming information form the sensors and generates output patterns which result in behavior. Additionally, the software evaluates the robot's performance by monitoring the output patterns and keeping track of potential bump events. The details of the evaluation method are described in detail in Section 2.2.

2.1.3 The control structure

The control structure of the robot in this experiment consists of an artificial neural network (ANN) that maps sensory input to motor output values. The ANN consists of a single layer with all-to-all connections, which means that every unit in the network receives input from every unit in the network, including itself. Of course certain connections may be effectively disabled by having a weight value of zero. The weight values of the network and the biases of the units are subject of evolution.

The network contains 10 units. Therefore it can be represented by a 10×10 matrix in which the weights of the connections between the units are stored and an array of size 10 which holds the biases of the units. The values of the two light sensors are treated as input to two of the units of the network. The output values of two other units are used to set the power of the motors. The value of the touch sensor is not fed into the network, but used only for determining the genotype fitness (see Section 2.2.1).

Learning does not take place during the life time of the network, so the weight and bias values remain constant for each individual. At each time step t the activation of each unit i is updated by summing the products of the activation all incoming units j and the corresponding weights w_{ji}. Each unit has a bias $bias_i$ which is added to this sum before the result is passed through a sigmoid function which puts the activations on a curve between $-.5$ and $.5$.

Input values from the light sensors are treated as activations coming into the network and are thus added to the net input of the corresponding input units.

2.1.4 The simulation

The Lego Mindstorms Simulator package (LMS, [6, 10]) was used for modeling the robot and its environment and to simulate its behavior. This simulation platform allows for detailed modeling of both the environment and the morphology of the the robot (see Figure 1(b) for a visualization). In addition, control structures developed for LeJOS [1, 3], an operating system for Mindstorms robots that allows for the execution of Java programs, can be used without modification in real robots as well as in robots simulated through LMS.

2.2 Evolutionary process

The evolutionary process applied to the control structure of the robot in this study is based on the ideas concerning genetic algorithms (GAs) as proposed by [4] and the work of [8] and [13] on the appliance of GAs to neural networks and that of [9] on Evolutionary Robotics.

The weights and biases to which evolution is applied were encoded into a genotype, constituted by an array of 110 floating point numbers: 100 representing the weights and for the bias values of a fully connected neural network consisting of 10 units. A control structure can be extracted easily by instantiating a neural network with the values from the array filled in properly.

There were 20 genotypes in each generation. The first generation consisted entirely of randomly generated genotypes. Subsequent generations contained a copy of the genotypes of the ten best performing individuals of the previous generation, complemented by *mutated* versions of the same ten genotypes. Mutation is a genetic operator proposed by [4]. It is essential for the evolutionary process, since it yields minor variations in the genotype, which may result in improved performance in the resulting individual. Within the mutated genotypes, each number was selected for mutation with a chance of $.05$. If a gene was selected for mutation, its value was altered by a random number from a gaussian distribution with $\mu = 0$ and $\sigma = .3$. These parameters are based on good results in preliminary experiments.

2.2.1 Fitness function

Selection of the individuals to be kept for following generations took place on basis of the *fitness* of each genotype. The fitness of a genotype was determined by assessing the performance of the resulting control structure. This process can be described by the following steps:

1. Fitness is set to zero
2. At each time step while the robot is operating:
   (a) If the robot is driving backwards, the square root of the product of the $power$ values of both motors is *subtracted* from the fitness; otherwise, if no bumps have occurred in the past 100 cycles, the square root of the product of the $power$ values of both motors is *added* to the fitness.
   The $power$ values of the motors are values that represent the speed at which the respective wheels are set to turn. Values of $power < 0$ indicate backward motion of the wheels. The $power$ values are calculated from the activations of the output units, resulting in values between -50 and 50.
   (b) If the bumper is touched, a penalty value is subtracted from the fitness. The penalty is a constant measure we set to the value of 50.
3. If $fitness < 0$: $fitness = 0$
4. Deliver fitness

The robot got to operate for a fixed number of time steps. In simulation mode, the number of steps was set to 500. The physical robot was programmed to report its fitness value after 300 time steps. This was done for reasons of efficiency. Since in this study it is not the intention to compare fitness values from the simulation and the physical environment, the difference is irrelevant.

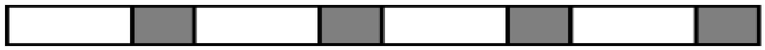
2.2.2 Comparability of real world fitness and simulation fitness

It is not the aim of this study to directly compare fitness values of the physical environment and fitness values of the simulated environment. Therefore, no measures were taken to guarantee the comparability of the fitnesses. Identically performing robots (assuming such a thing exists) in different environments (one in simulation, one in the real world) are unlikely to obtain even remotely comparable fitness values. An important cause for this difference, is the difference in the number of time steps between environments. Furthermore, there are numerous factors beyond verification of control that might influence the fitness measurements and are likely to differ between environments. Therefore, and since no intent is made to make comparisons between environments, one should consider ‘real world fitnesses’ and ‘simulation fitnesses’ to be on independent and incomparable scales.

2.3 Conditions

The experiment consisted of two conditions, in each of which a control structure was evolved over 55 generations, either fully in simulation or in an interleaved fashion, followed by five final generations of evolution in the real world. Only one run was executed for each condition due to time restrictions.

(a) Condition 1: fine tuning



(b) Condition 2: interleaving

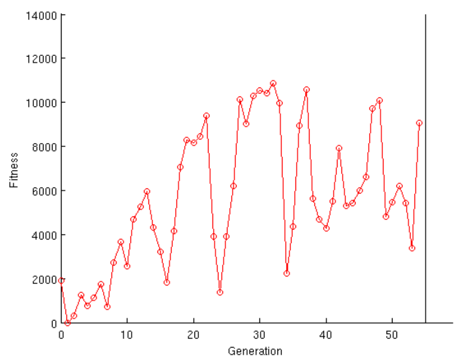
Figure 2: Schematic presentation of the two conditions of the experiment. The bars represent the entire course of evolution. The white zones stand for evolution in simulation; the grey parts represent evolution in the physical environment.

The results of the experiment are based on a comparison of the performance of the robot in the five final generations in both conditions. For a schematic overview of both conditions, see Figure 2.

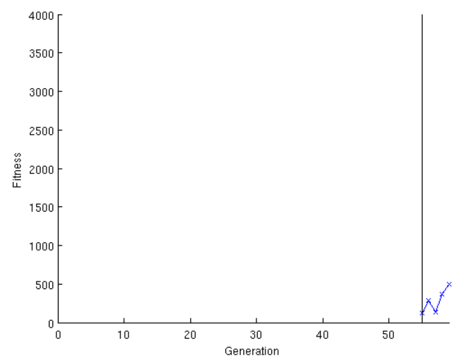
2.3.1 Complete simulation

In the first condition, the control structure was evolved in simulation over the first 55 generations. Five generations of real world evolution followed. This method is similar to the one used for example by [7].

Computation of the Pearson coefficient showed a reasonable correlation between generation and fitness for the first 55 generations ($r = .27$) as well as for the final five ($r = .24$), which indicates that the fitness has improved over the course of evolution, in both simulation and real world.



(a) Simulation



(b) Real world

Figure 3: Fitnesses for the ‘fine tune’ condition. Notice that the fitness measures in the physical environment and those in the simulated environment are *not comparable* (see Section 2.2.2 for an explanation).

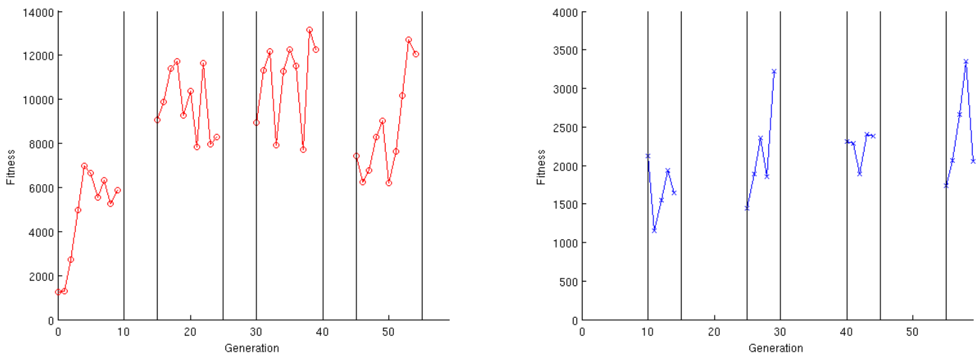
2.3.2 Interleaved evolution

In the second condition, epochs of evolution in simulation and evolution in the real world were alternated. The first ten generations were evolved in simulation, followed by a series of five generation evolved in the physical environment. This pattern was repeated until 60 subsequent generations had been evolved in total, of which 40 were the result of simulated evolution and 20 were evolved in the real world.

In this condition, reasonable correlations were found as well. For the entire set of generations evolved in simulation, a correlation of $r = .35$ between generation and fitness was found; the generations evolved in the physical environment yielded a correlation of $r = .20$.

2.4 Results

A robot control structure was evolved in two conditions. In the first condition, evolution in simulation and evolution in a physical robot were interleaved during the first 55 generations. In the second condition, the control structure was evolved in simulation entirely over the first 55 generations. To test whether interleaving simulation and real world evolution affects the performance of an eventual control structure for a physical robot, five final generations were evolved in a physical robot in both conditions.



(a) Simulation (b) Real world

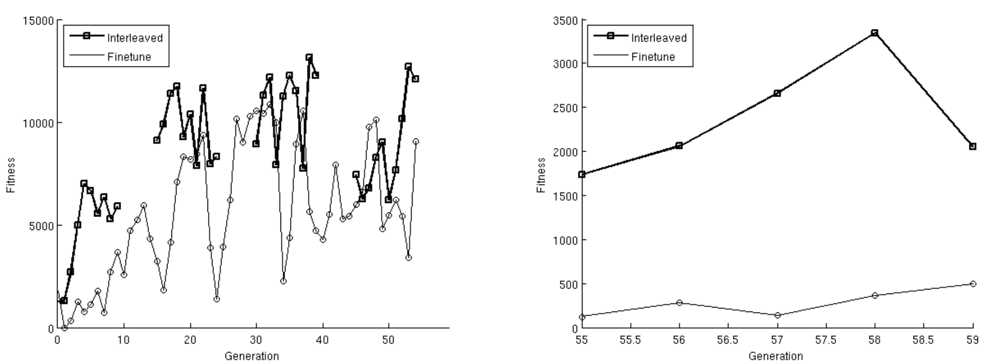
Figure 4: Fitnesses for the ‘interleaved’ condition. The values are the average fitnesses of the ten best performing individuals per generation.

Condition	Fitness N	Mean	SD
Fine tune	100	142.3	260.6
Interleaved	100	1586.1	1067.0

Table 1: Descriptives

The fitnesses of the genotypes from these final generations were compared. As can be seen in Table 1, the mean fitness of the ‘interleaved’ condition over the final generations is about ten times as high as that of the ‘fine tune’ condition. In Figure 5(b) one can see that the average fitness of the ‘interleaved’ condition is in fact higher in each of the final generations. An analysis of variance over the final five generations showed that the observed difference between conditions is significant ($F(1, 198) = 172.76, p < .01$).

This means interleaving simulation and real world over the course of evolution positively affects the performance of the eventual control structure. The strength of the effect is reasonable ($\eta^2 = .28$).



(a) Simulation (b) Real world, final generations

Figure 5: Fitnesses for both conditions. The dark line shows the average value of the ten best performing individuals for each generation in the ‘interleaved’ condition; the thin line represents the ‘fine tune’ condition.

2.5 Conclusion

The results of the experiment described above show that interleaving simulation and a physical environment over the course of evolving a control structure for a physical robot positively affects the fitness of control structures subsequently evolved in a physical robot. These results support the claim that a control structure

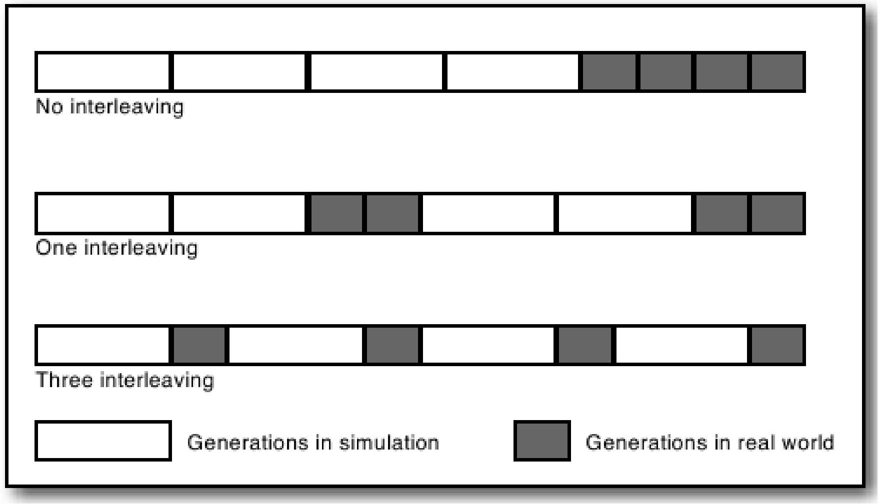


Figure 6: Proposed conditions for the new experiment.

in evolution can be prepared for its eventual habitat from the early stages of evolution on. However, it cannot be concluded that the results are either partially or completely due to the interleaving per se. This issue is discussed in the following section.

3 Discussion

We showed that moving epochs of the evolutionary process from simulation to real world improves the quality of the resulting control structures. This effect could be explained in two non-excluding ways. First, the interleaving itself may have beneficial properties, as hypothesized in this study. Second, as can be seen clearly in Figure 2, the sheer amount of generations evolved in the real world differs vastly between conditions, which may strongly contribute to the effect.

In a study currently being conducted by one of the authors, the effect of interleaving is assessed in an experiment in which the ratio between 'real world' and 'simulation' generations remains equal (see Figure 6 for a schematic overview). In contrast, the *distribution* of the fixed number of real world generation is varied over conditions. As a consequence, the length of each epochs also differs between conditions. Depending on the outcome of this new study, the effect of epoch length may also have to be investigated.

References

[1] B. Bagnall. *Core LEGO MINDSTORMS Programming: Unleash the Power of the Java Platform*. Prentice Hall, Upper Saddle River, NJ, 2002.

[2] P. Eggenberger Hotz and G. Gómez. The transfer problem from simulation to the real world in artificial evolution. In M. Bedau, P. Husbands, T. Hutton, S. Kumar, and H. Suzuki, editors, *Workshop and Tutorial Proceedings of the Ninth International Conference on the Simulation and Synthesis of Living Systems (Alife IX)*, pages 17–20, 2004.

[3] G. Ferrari, A. Gombos, S. Hilmer, J. Stuber, M. Porter, J. Waldinger, and D. Laverde. *Programming LEGOMindstorms with Java*. Syngress Publishing, Rockland, MA, 2002.

[4] J. H. Holland. *Adaptation in Natural and Artificial Systems*. MIT Press, Cambridge, MA, 1975.

[5] N. Jacobi, P. Husbands, and I. Harvey. Noise and the reality gap: The use of simulation in evolutionary robotics. In *Proceedings of the Third European Conference on Advances in Artificial Life*, pages 704–720, London, UK, 1995. Springer-Verlag.

[6] B. Künsting. Konzeption und prototypenhafte implementation einer lego-mindstorms simulationsumgebung - lejos-emulation. Master's thesis, Universität Paderborn, 2004.

[7] O. Miglino, H.H. Lund, and S. Nolfi. Evolving mobile robots in simulated and real environments. *Artificial Life*, 2(4):417 – 434, 1995.

[8] M. Mitchell. *An Introduction To Genetic Algorithms*. MIT Press, Cambridge, MA, 1996.

[9] S. Nolfi and D. Floreano. *Evolutionary Robotics*. MIT Press, Cambridge, MA, 2000.

[10] W. Sträter. Konzeption und prototypenhafte implementation einer lego-mindstorms simulationsumgebung - simulationsmaschine. Master's thesis, Universität Paderborn, 2004.

[11] J. Walker, S. Garrett, and M. Wilson. Evolving controllers for real robots: A survey of the literature. *Adaptive Behavior*, 11(3):179 – 203, 2003.

[12] M.S. Wilson, C.M. King, and J.E. Hunt. Evolving hierarchical robot behaviours. *Robotics and Autonomous Systems*, 22:215 – 230, 1997.

[13] X. Yao. Evolving artificial neural networks. *Proceedings of the IEEE*, 87:1423 – 1447, 1999.