# The Search for a Bridge Over the Reality Gap

## Effects of Interleaving Interval Duration and Quantity

By: Rik van den Brule
Affiliation: Department of Artificial Intelligence, Radboud University Nijmegen
Student number: s0425842
Email: r.vandenbrule@student.ru.nl
First Reader: Pim Haselager
Second Reader: Ida Sprinkhuizen-Kuyper
Date: January 21, 2008

# Contents

## Abstract

In Evolutionary Robotics, simulators are used to speed up the lengthy process of evolving control structures for real world robots. The downside of this approach is that control structures evolved solely in simulation tend to perform not as good in the real world as they did in the simulation. This drop in performance has come to be known as the 'reality gap'. Interleaving simulation with real world in the evolutionary process is a newly proposed method which might help to bridge this gap, but it lacks solid empirical proof. In this study, a systematical experiment was done in which the same proportion of generations was differed in the amount of interleavement over three conditions. Interleaving was found to have no significant effect on the final fitness values of the control structures. However, the amount of *consecutive* real world generations was found to have a positive effect on the final fitness. Unfortunately, it is hard to reach definite conclusions because of time constraints involved with a Bachelor's Thesis and the stochasticity of the evolutionary algorithm.

## Acknowledgments

I would like to thank Twan Goosen, Pim Haselager, Ida Sprinkhuizen-Kuyper and Joris Janssen, and all the others for their input and feedback and for the enlightening discussions we have had. A special thank you to everyone who was patient enough to let me explain my thesis to them, which helped me to boil it down to the bare necessities and helped me to find the bigger picture when I got stranded in the details.

# Chapter 1

# Introduction

In this chapter, a brief introduction of evolutionary robotics is given. Next, the Reality Gap, one of the major unsolved problems in this field of research, is explained, along with some proposed solutions. One of such proposed solutions, dubbed the 'Interleaving method', is explained in more detail since this thesis is based on that work. Finally, the research questions are addressed.

## 1.1 Evolutionary robotics

Evolutionary robotics is a field of robotics in which principles of Darwinian evolution are used to develop autonomous robots automatically (Nolfi & Floreano, 2000). Evolutionary algorithms, first proposed by Holland (1975), are generally used in this approach. These algoritms simulate the process of selective reproduction of the individuals with the best behavior in a large population.

One aspect of this approach is that the behavior of every individual in the population needs to be evaluated. This is by far the most time consuming step of the evolutionary process. Running this step of the evolutionary process on a physical robot in real time is generally prohibitive because of time constraints (Walker, Garrett, & Wildon, 2003). However, this approach has been used (for example, Floreano and Mondada (1994)), and it lead to robust results. Unfortunately, it took 10 consecutive days of training to evolve simple homing behavior for the robot in this study. Evidently, this approach is very time consuming, and therefore there has been a trend towards the use of simulators in evolutionary robotics.

There are two main advantages to using a simulator above a real world robot. First, a simulator is faster than a robot in the real world. This way, the evaluation of the individuals, which is the greatest bottleneck in this kind of research, can be sped up considerably. Second, it is not necessary to have someone present during the evolutionary process since the process can be better automatized in a simulator. A simulator can be set up to run a virtually unlimited number of times consecutively, while in the real world, someone needs to place the robot at its starting point, and depending on the hardware used, needs to upload a new control structure for every trial.

## 1.2 The reality gap

The main disadvantage when using a simulator for the evolutionary process is that a simulator is only a model of the real world. Therefore, when a control structure evolved in a simulator is directly ported to a real robot, a drop in performance should be expected. This 'reality gap' has proven to be hard to bridge. A number of solutions have been proposed to solve this problem, each of which yield acceptable results. However, the definite answer has not yet been found.

Nolfi, Floreano, Miglino, and Mondada (1994) reviewed studies in which robots were evolved. Unfortunately, they do not present an objective way of comparing the different possible strategies presented. One of the studies they described used the simulator only for the evolutionary phase of the experiment (Miglino, Nafasi, & Taylor, 1994). Efficient wandering behavior for small robots was evolved. The evolutionary process took about three hours to complete. Only three individuals (each from different generations) were 'embodied' in a real robot and tested. The behavior in the embodied phase was similar to that in simulation, although the higher noise of the real world altered the expected trajectory of the robot somewhat. Overfitting of the behavior to the much cleaner simulation environment also became a problem during later phases of development, which led to a drop in performance. An other experiment, by Nolfi, Miglino, and Parisi (1994) suggest a 'fine tune' phase in the evolutionary process to let the control structures get used to the real world after evolution in simulation. The first 300 generations of the evolutionary process were done in simulation, followed by another 30 generations of evolution in the real word. The first 300 generations took only one hour to complete, but the 30 real world generations must have taken a significant amount of time, as noted by Walker et al. (2003). When tested in the real world, straight-out-the-simulator control structures performed significantly worse than in the simulator. After the fine tune phase, fitness levels in simulation and in the real world were of comparable value. It should be noted that the simulator used in this study was created by sampling the real environment with the sensors of the robot used, which led to a very accurate simulation of the environment. This means the simulator was a very accurate model of the robot's view of the world. The authors did this to make the transition to the real world easier for the control structures, but an initial performance drop of almost 50% was still reported.

## 1.3 Interleaving

Goosen (2007) (see also: Goosen, van den Brule, Janssen, and Haselager (2007)) describes an experiment which compares different ways of evolving the behavior of simple robots in simulation and the real world. He used an evolutionary algorithm to develop control structures (an Artificial Neural Network, or ANN) for simple Lego Mindstorms robots which had to perform a simple task. In the case of this experiment, robots had to travel as far as possible without bumping into walls, which were marked by a black area around them on the floor. The robots had two light sensors mounted on them, pointing downwards, so the robots could detect light and dark areas on the floor. A good control structure can avoid walls by using this information.

The experiment Goosen (2007) describes consisted of a between-subject design with two conditions. The control, or fine tune condition lets control structures evolve for 55 generations in simulation, after which the control structures were evolved for another five generations on a real

world robot. The other condition used a different technique, which he called *interleaving*. In this condition, the simulation is interrupted after 10 generations. The control structures thus far evolved in simulations are then placed in a real world robot, and the control structures are evolved for 5 generations in the real world. The resulting control structures are then placed in the simulator again for further evolution. This process is repeated four times (for a graphical overview, see Figure 1.1).

It is believed that interleaving real world trials with simulation, control structures are being prepared for the real world in the early stages of their evolution. This allows them to perform better than controls after evolution and prevents overfitting to the simulation.

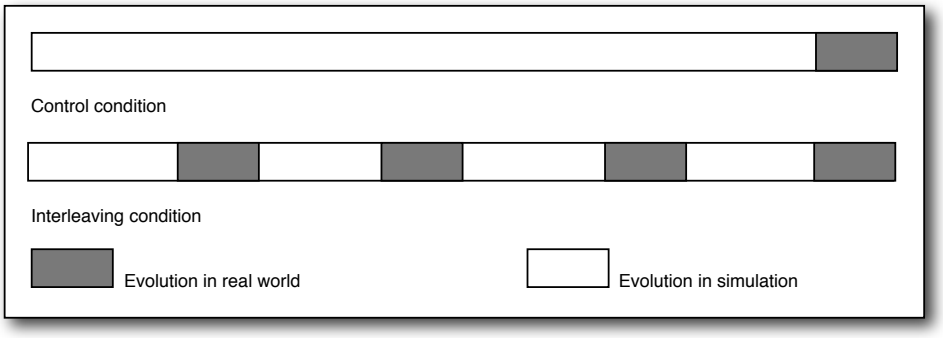



Figure 1.1: Conditions in the experiment conducted by Goosen (2007).

Goosen (2007) found that, although fitness improves over time in both conditions, the final five generations of individuals in the interleaving condition perform better than those of the control condition. Unfortunately, because of the setup of this experiment, the higher fitness of the individuals in the interleaving condition cannot be attributed solely to the effect of alternating simulation with real world generations. Since the individuals of the interleaving condition are also being exposed more to the real world, the difference in fitness could also be caused by this factor.

Another point is that Goosen (2007) based his conclusions on the analysis of fitness of the final five generations of each condition. This might lead to errors since the control structures are still evolving at this time. It is better to conduct special test trials of the best individuals of the final generation of each condition, and base the analysis on those.

## 1.4 Problem statement

The experiment described in this Bachelor Thesis continues the work of Goosen (2007). Its setup is designed to be similar to that of Goosen, while it tries to avoid the weak parts of the study. The basic research question is how much of the effect found by Goosen (2007) can be attributed to the real world exposure time, and how much of it is caused by interleaving real world generations within the simulation phase of the experiment.

A second question is whether the control structures produce behaviour which is *reliable* (i.e. produces the same behaviour every time). Because of the selected method for analysis, it is possible to see whether this is the case or not.

In Section 2, the setup of this experiment is described in detail. Results are presented in

Section 3, which are discussed in Section 4. Section 5 contains the conclusion and areas in which further research can be undertaken.

# Chapter 2

# Method

The experimental setup of this research is described in this Chapter. The Task, environment and robot are described in Section 2.1. An overview of the differences between the simulator and the real world is given in Section 2.2. Section 2.3 explains the evolutionary process and the parameter settings of the algorithm. In Section 2.4, the three conditions in the experiment are explained, while Section 2.5 deals with the possible predicted outcomes of the experiment and what they imply for the interleaving effect.

## 2.1 Task

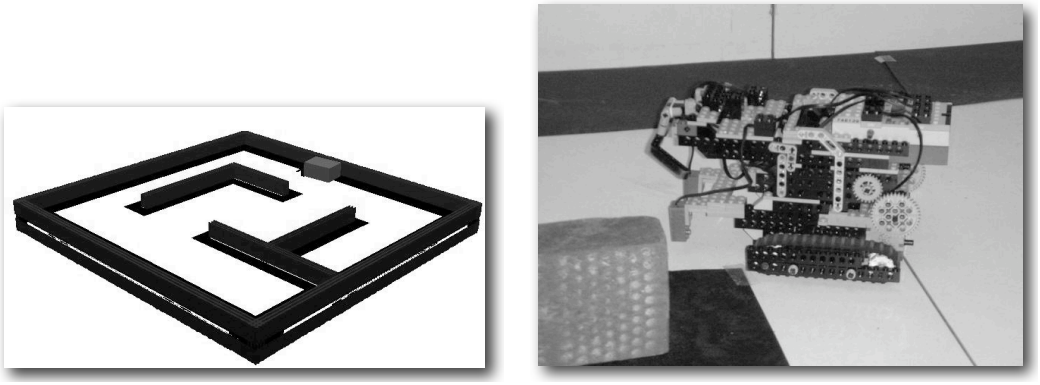
To keep the experiment similar to that of Goosen (2007), most of the task and equipment were unchanged from the experiment described in his thesis. The task the robot is given is to travel as far as possible without bumping into the walls of the environment within a given number of cycles. The robot is placed at the same starting position at each run.

### 2.1.1 Environment

The environment in which the robot has to perform has a white floor and is walled off at all sides. Some walls are also placed inside the environment (see Figure 2.1(a). The floor was made of white tiles, which were covered with strips of black paper around the walls. The difference of light intensity between the white and black parts of the floor is big enough to be picked up by the light sensors used for the robot. The walls around the environment were made of the same white-coated chipboard as the floor, and bricks were used for the interior walls. This environment design makes it possible for a good control structure to predict whether it is close to a wall by monitoring the light intensity, and avoid a wall by turning away.

### 2.1.2 Robot

For trials in the real world, the Lego Mindstorms robot created by Goosen (2007) was used (see Figure 2.1(b). The robot has two light sensors mounted on the front of the robot, pointing to the floor, and a front bumper. The light values measured by the light sensors are fed into the control structure (see Section 2.1.4 for details on the control structure). The bumper is only used



(a) environment (b) robot

Figure 2.1: The layout of the environment, simulated view (a). The robot in its environment (b). Adapted from Goosen (2007)

to detect a collision, and is not used by the control structure to drive the robot. A Lego motor on each side of the robot powers a caterpillar track. If both tracks turn forward or backward at the same time, the robot will travel in a straight line. If one of the tracks goes forward while the other is turning backwards, the robot will make a turn.

### 2.1.3 Simulator

The Lego Mindstorms Simulator (LMS) (Künsting, 2004; Sträter, 2004), enhanced with a script to allow for multiple runs needed for the evolutionary process created by Goosen (2007), was used in this experiment. LMS can model both an environment and a Lego robot. It also has the advantage that it can simulate the 'brick' controller using the LeJOS operating system, so that programs (i.e. control structures) written in Java can be exchanged between the simulator and the real world robot. The same virtual representation of the robot and the environment created by Goosen (2007) were used for the simulator.

### 2.1.4 Control structure

The same artificial neural network (ANN) used in Goosen's study (2007) was used. The 10 units of this single layer network all have connections to every other unit in the network. Each unit also has a bias weight. The values of the light sensors are added to the input of on two units, while the output of two other units is used as control signals to the motors.

The activation of each unit is updated every cycle $t$ of the run. A new value is first computed by one of the two different formulae (one for units which receive input from a light sensor and one for the ones who do not). The resulting values from these formulae are then scaled to a sigmoid curve scaled between $-.5$ and .5 by the following function:

$$scaledActivation_{i,t} = \frac{1}{1 + e^{-activation_{i,t}}} - .5$$

The activation function of units which do not use input from a light sensor directly is the

following:

$$activation_{i,t} = bias_i + \sum weight_{i,j} \times activation_{j,t-1}$$

The two units that do use the input from a light sensor use a slightly different activation function, defined as:

$$activation_{i,t} = bias_i + \sum weight_{i,j} \times activation_{j,t-1} + sensor_{i,t}$$

In order to use an ANN in a genetic algorithm, it needs a representation which can be easily manipulated by the evolutionary process (see Section 2.3). In other words, it needs a genetic code. The structure of the network used is fixed (i.e., no connections are added or deleted during the evolutionary process), so only the weights of the connection between units needs to be represented by this code. This can be achieved by representing all weights and biases as an array of floating point numbers, 110 in total.

## 2.2 Differences between simulator and real world

Because of the time constraints of this bachelor project, a few changes needed to be made to the real world runs in respect to the simulator runs to complete the experiment in time. Since the real world generations are the most time consuming and labour intensive of the experiment, it was the most effective place to make changes.

First of all, like Goosen (2007), the number of cycles in each run for the simulator and the real world are different. In the simulator, the robot gets 500 cycles to drive around in the environment. In the real world, it only gets 300 cycles. Second, in the simulator, every control structure was tested twice, and the fitness values of the two runs averaged. While this improved accuracy of the reported fitness, it proved to be too time consuming to be applied in the real world. Note that Goosen (2007) tested control structures only once, both in simulation and the real world.

This experimental setup does not allow for a direct comparison between fitness attained in the simulator and fitness attained in the real world. As Goosen (2007) states: "...one should consider 'real world fitness' and 'simulation fitness' to be on independent and incompatible scales."

## 2.3 Evolutionary Process

The evolutionary algorithm used in this study is the same as Goosen (2007), who in turn based his approach on the work of Nolfi and Floreano (2000) on Evolutionary Robotics. Also used was the work of Holland (1975) and Mitchell (1996) on evolutionary and genetic algorithms, and finally Yao (1999) for application of these algorithms on neural networks.

### 2.3.1 Parameters

All parameters of the evolutionary algorithm used in this study are copied from the work of Goosen (2007), and are summarized in Table 2.1. Every generation consists of 20 individuals. The ten best individuals are selected for the next generation and their offspring replaces the ten worst performing individuals of the previous generation. The new individuals are created by mutating

the ten best individuals. A gene has a chance of .05 to mutate. This results in an average of 5 to 6 mutations per genome.[1] A gene is mutated by changing its value with a random number, selected from a Gaussian distribution with $\mu = 0$ and $\sigma = .3$. No crossover operator was used since Yao (1999) states that "[the crossover operator] does not perform well in searching for near-optimal ANN architecture" (pp. 1425).

| Parameter name | Parameter value |
|---|---|
| population size | 20 |
| number of offspring | 10 |
| mutation probability | .05 per gene |
| crossover probability | 0.00 |

Table 2.1: Settings of free parameters of the evolutionary algorithm used in this study, as chosen by Goosen (2007) and used in this study.

### 2.3.2 Fitness function

Performance of a control structure is measured by a fitness value, which is a combination of both distance traveled during the run and the number of bumps recorded by the front bumper. The distance measure is calculated from the output of the control structure to the motors according to the following formula:

$$output_{left} < 0 \wedge output_{right} < 0 \Rightarrow \Delta fitness = -\sqrt{output_{left} \times output_{right}}$$

$$output_{left} > 0 \wedge output_{right} > 0 \Rightarrow \Delta fitness = \sqrt{output_{left} \times output_{right}}$$

$$allothercases \Rightarrow \Delta fitness = 0$$

This set of formulas give the control structure an increase in overall fitness when it moves forward, and a decrease in overall fitness when it moves backwards. When the robot makes a turn or is standing still, it will not gain or lose any points.

The number of bumps during the run is counted. For each bump, a penalty of 50 points is subtracted from the total fitness value. The formula for the total fitness is:

$$fitness = \sum_{i=1}^{n_{cycles}} \Delta fitness_i - 50 \times n_{bumps}$$

An important difference made to the fitness measure as used by Goosen (2007) is that negative fitness values are allowed in this study. Goosen decided against negative fitness in his experiment, and instead reported them as 0. Because of this, it is not possible to distinguish a control structure which does not move the robot at all from a control structure which causes the robot to go

[1] Each gene in the genome of 110 genes has a probability of 0.05 of being mutated. The expected number of mutations per genome therefore is $110 \times .05 = 5.5$

backwards in Goosen's experiment. Although negative fitness values are not commonplace in evolutionary algorithms, they are useful to distinguish bad control structures from the *really* bad ones (i.e., that show behavior opposite to selective advantages). Therefore, the decision was made to allow negative fitness values in this experiment.

## 2.4 Conditions

As a pilot study, a single run of 76 generations in simulation was conducted to see if this number of generations is sufficient to reach a plateau in fitness, i.e. to see whether the fitness reaches a maximum in the number of generations run in this experiment. Each robot was evaluated once. The results of this pilot can be seen in Figure 2.2.

Two observations can be made after analyzing the data of this pilot experiment. First, a positive correlation was found of average fitness over generations ($r = 0.776$, $p < .0001$). This shows that the mean fitness of the population improves over time. Second, the maximum and mean fitness of the population stops improving at around 40 generations. This indicates that a maximum of fitness is reached. Thus, an evolutionary process of 52 generations, as used in the experiment (see below), should also be sufficient to reach an optimum in fitness value.

Three conditions are used in the experiment. The number of generations in the real world (12) and in simulation (40) is equal in all three conditions. The conditions only differ from each other in the duration and position of the real world generations during the evolutionary process. A graphical overview of this distribution per condition is presented in Figure 2.3.

In the no-interleaving condition, the robots are evolved in the simulator for 40 generations and are then further evolved in the real world for another 12 generations. One could say this condition has a 40/12 structure. In the one-interleaving condition, the simulation and real world generations are split in two. Now, the robots will be evolved for 6 generations in the real world after just 20 generations in the simulator (which makes the structure 20/6). This is repeated twice. In effect, this results in a condition which has 6 real world generations interleaved within 40 generations in simulation, completed by another 6 generations in the real world. For the three-interleaving condition, the 20/6 structure is divided again as described above, leading to two 10/3 structures. Four 10/3 structures are ran after each other, which in effect gives this condition three real world interleavings of 3 generations each, at a quarter, halfway and three quarters in the simulation training, followed by an additional 3 generations in the real world.

This design eliminates the effect more real world generations may have on the fitness level of the final generation, something Goosen (2007) did not keep constant in his research, since the number of real world and simulated generations are the same in each condition.

After the evolutionary process is completed, the best 10 individuals will be selected from the final generation of each condition. This is done because the mutation operator might have caused mutations that result in bad control structures. These are normally filtered out and overwritten by offspring of the best 10 individuals by the evolutionary process. Thus, for comparing the different conditions, only the best 10 control structures need to be tested. These individuals will be run once more in the real world to make sure the fitness of these individuals was not subject to (un)lucky runs.

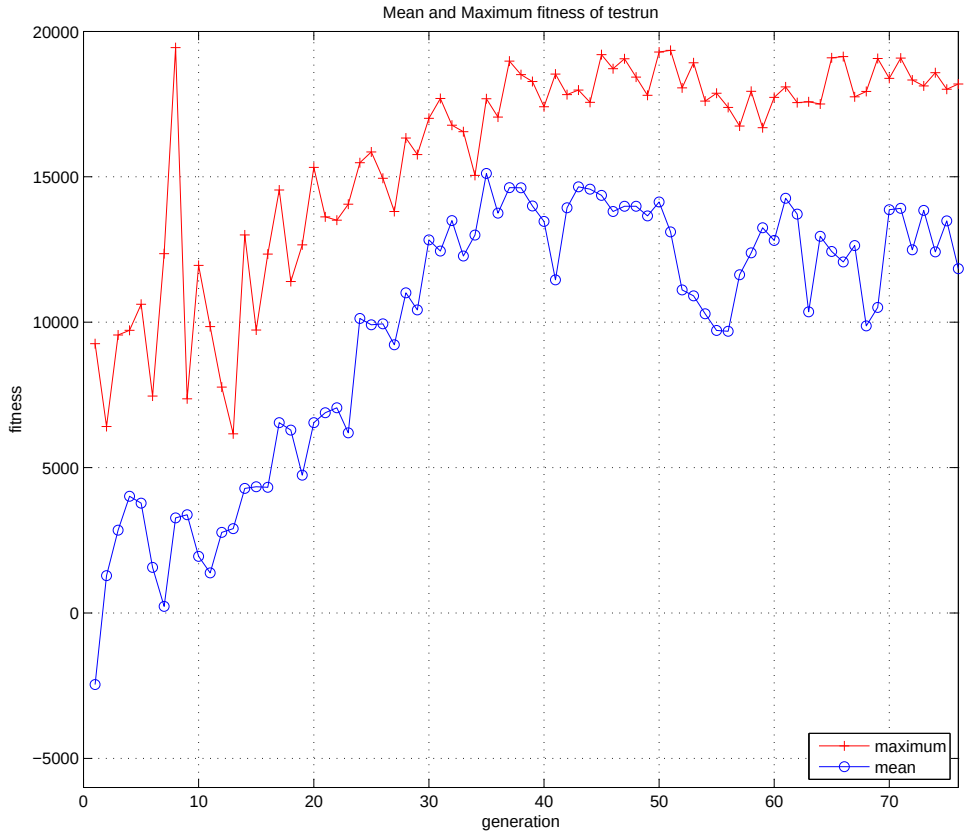



Figure 2.2: Results of the test run to validate the number of generations used in the real experiment.

## 2.5 Number of runs

Because of time constraints associated with a Bachelor Thesis, there was only time to perform a single run of the experiment. Because evolutionary algorithms are stochastic in nature, an element of chance has influence on the results. This limits the conclusions that can be drawn upon this work. The most ideal situation would be to do enough runs to make a *T-test* possible (a sample size of approximately 20). This *t-test* should be done on the best individual of each generation, and would provide solid statistical evidence despite the stochastic nature of the algorithm.

Since running the experiment 20 times was not possible due to time limitations, the decision was made to base the analysis of the results on the mean of the best 10 individuals of each generation. An analysis on only one individual per generation of one run would bias the results too much. Taking the individuals which are selected for reproduction is an adequate measure of the 'goodness' of a generation, because only the improved mutations are selected for the analysis, and bad mutations are left out since they will not perform better than their parents.

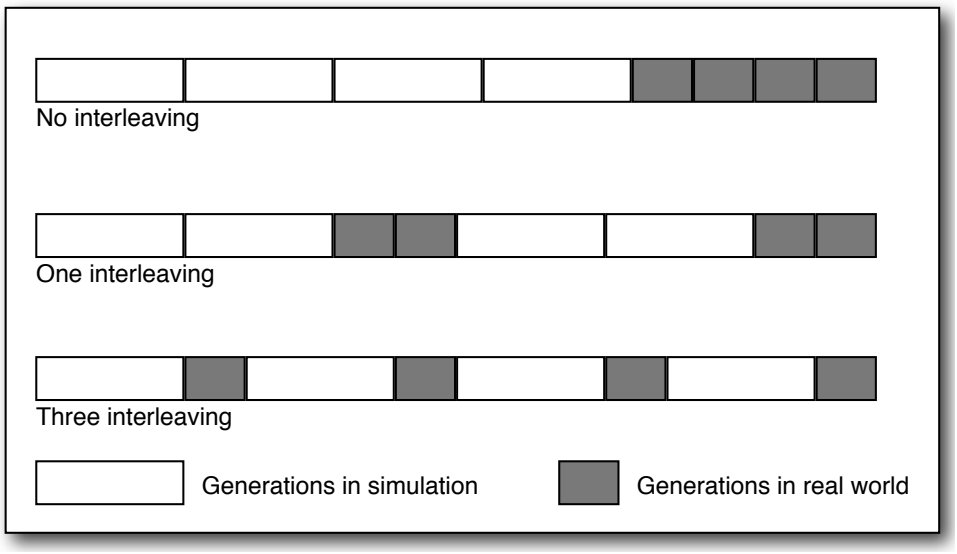



Figure 2.3: Distribution of generations in simulation and the real world in different conditions of the experiment.

## 2.6 Predictions

There are several possible outcomes of the experiment, depending on the mechanisms involved in the interleaving effect. As noted before, the effect found by Goosen (2007) might be caused by interleaving, total real world exposure time, or both. Five different outcomes of the experiment are possible, listed below. The first four predictions deal with the ways interleaving or real world exposure contribute to the difference in mean fitness of the final generations, while the fifth prediction has to do with the duration of the real world exposure time.

1. Effect of interleaving, no effect of total real world exposure. If there is only an effect of interleaving on the final fitness, and no effect of the real world exposure, the effect of the three interleaving condition will be the largest, followed by the one interleaving condition, and finally the no interleaving condition. In short, the result of the experiment will be:

   $\mu_{no\_interleaving} < \mu_{one\_interleaving} < \mu_{three\_interleaving}$

2. No effect of interleaving, effect of total real world exposure. If this is the case, the mean fitness of the conditions in this experiment should all be equal to each other, since all conditions have the same number of generations in simulation and the real world evolution. In short:

   $\mu_{no\_interleaving} = \mu_{one\_interleaving} = \mu_{three\_interleaving}$

3. Effect of interleaving, effect of total exposure. If both variables play a part in the effect, the results of the experiment will be similar to the results when only interleaving has an effect on mean fitness, but the difference of mean final fitness between the conditions will be smaller. This is a bit of a problematic point, since there would be no way to compare this difference. A follow-up experiment will be required to distinguish between these possible outcomes, but a comparison to the results of Goosen (2007) could also shed some light on the results. The difference in mean average fitness can be described as:

$\mu_{no\_interleaving} < \mu_{one\_interleaving} < \mu_{three\_interleaving}$

4. No effect of interleaving, no effect of total real world exposure. It is possible that the effect measured by Goosen (2007) was coincidental, and none of the factors were of influence on the final fitness. An unknown variable, not controlled in the experiment, might be the cause of the measured effect.

5. The *duration* of continuous real world exposure time could also have an effect on the mean final fitness. In that case, the mean final fitness of the condition with no interleaving will be the highest, because it has twelve consecutive generations in the real world. This condition is followed by the one-interleaving condition, and finally the three-interleaving condition since it has the fewest consecutive number of real world generations. Hence, the outcome of the experiment will be:

   $\mu_{no\_interleaving} > \mu_{one\_interleaving} > \mu_{three\_interleaving}$

In Section 4, the outcome of the experiment is discussed in relation to these predictions.

# Chapter 3

# Results

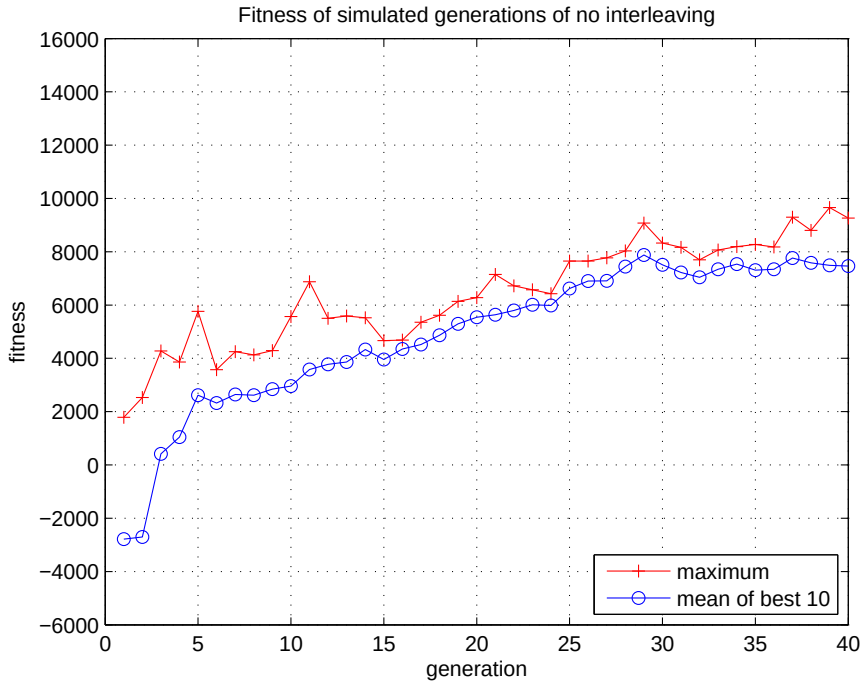
In this section, the results of the experiment are discussed. Section 3.1 deals with the evolution over time in every condition separately. The conditions themselves are compared in Section 3.2.

## 3.1 Evolution in conditions

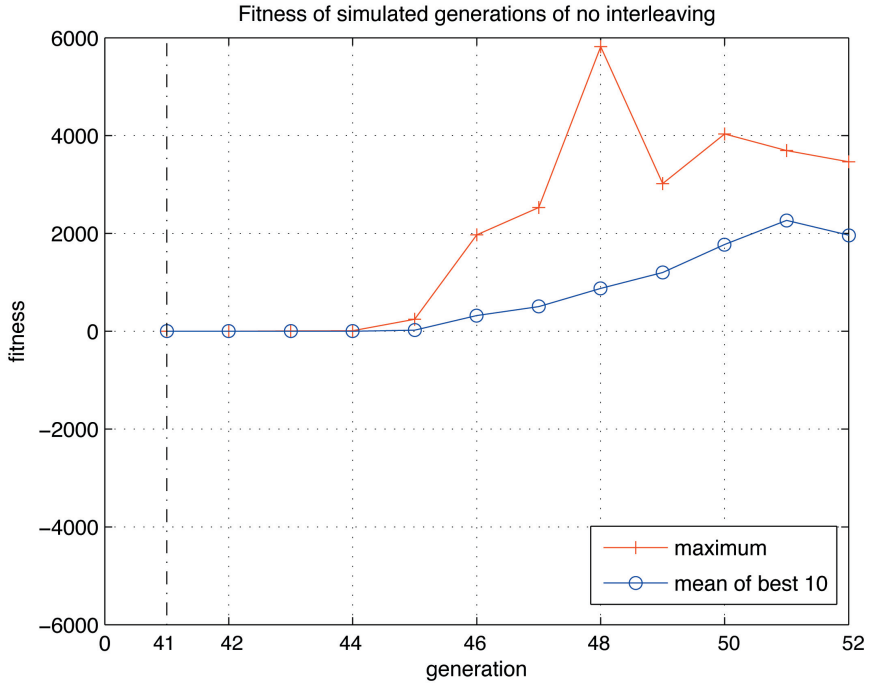
Figures 3.1, 3.2 and 3.3 show the fitness values of both the generations in simulation (subfigures *a*) and real world (subfigures *b*). As mentioned in Section 2.5, the analysis of all results are based on the best ten individuals of each generation. Each figure shows the fitness of the best individual as a red line. The blue line is the average of the ten best individual of the generation. When the condition is switched to the real world and back, two vertical black lines are shown to indicate this break. This is done because the fitness levels between the real word and the simulator are not to be compared, so it was not possible to present both simulation and real world fitness results in one figure.

A linear fit of the data shows that there is a positive correlation between mean fitness and generations in most of the conditions in simulation and the real world (Table 3.1), with the exception of the real world generations of the three-interleaving condition ($r^2 = .4151$, $p < .0001$).

It can clearly be seen that, after each time control structures are taken out of the simulator, their performance drops considerably, in some cases even below 0. Fitness values in simulation are clearly much higher than those observed in the real world.

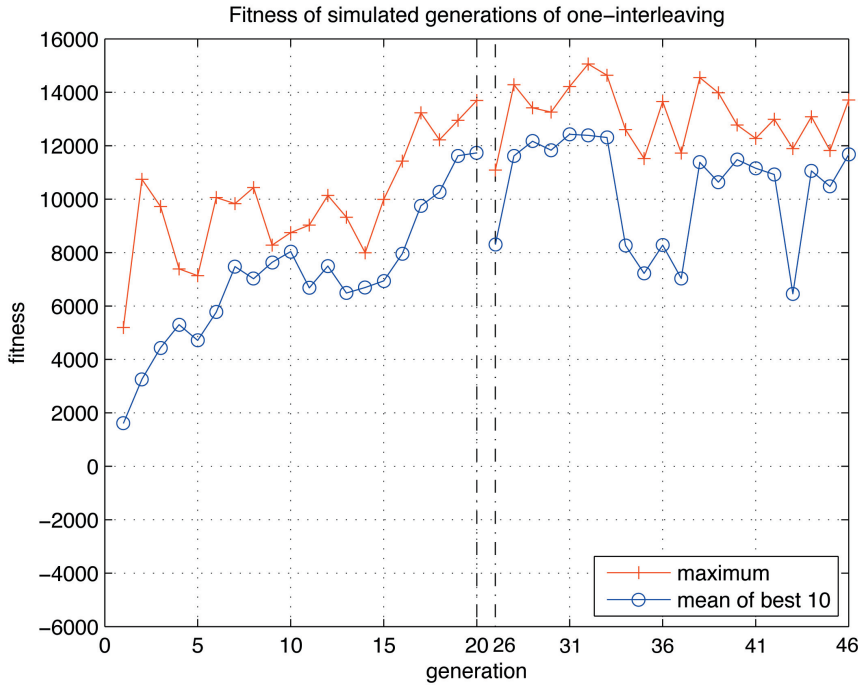



(a) simulation

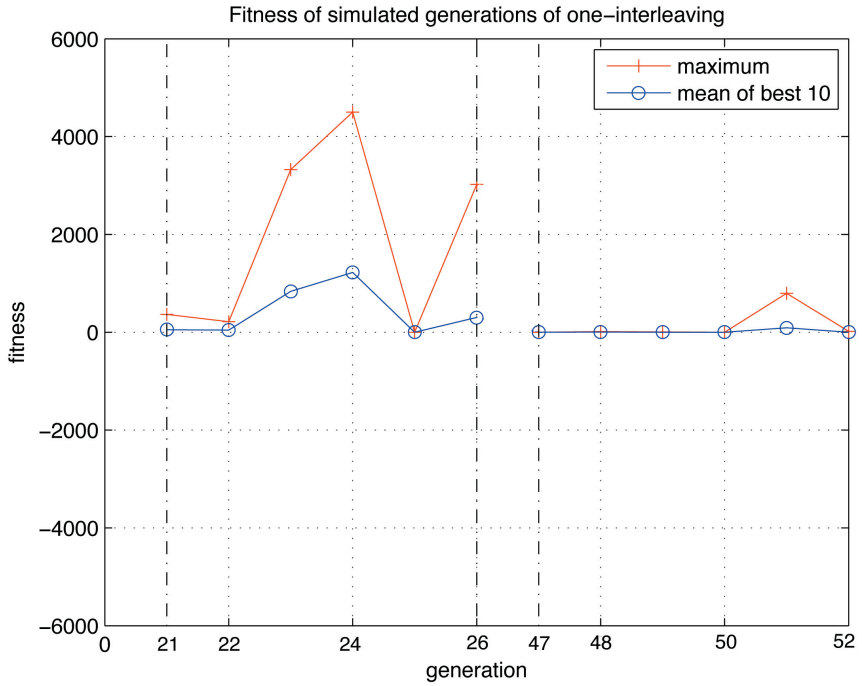



(b) real world

Figure 3.1: Fitness development in simulation (a) and real world (b) generations for the no-interleaving condition.

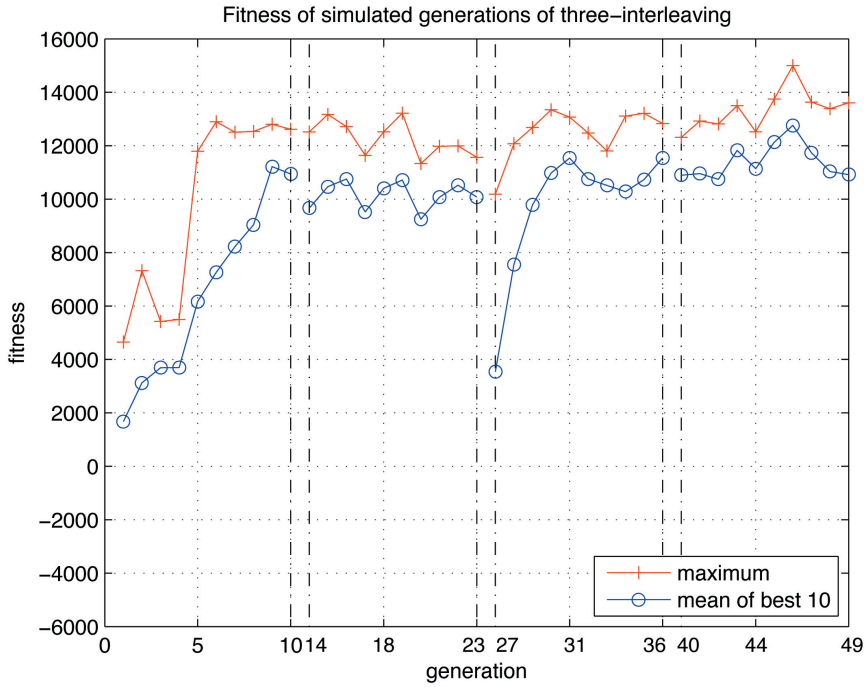



(a) simulation

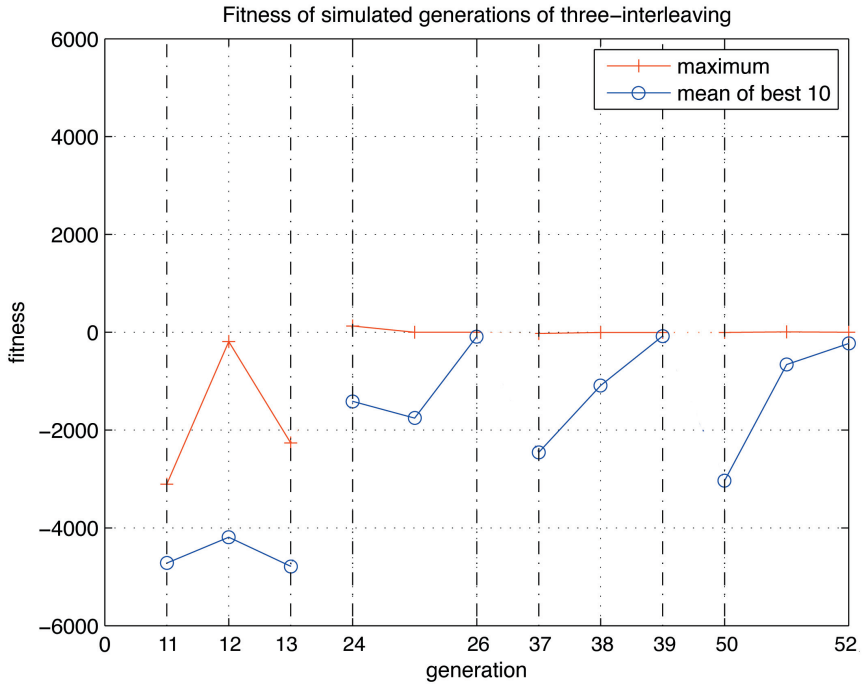



(b) real world

Figure 3.2: Fitness development in simulation (a) and real world (b) generations for the one-interleaving condition. The vertical dotted lines indicate where the chain of generations in the simulator or real world have been interrupted by an interleaving.




(a) simulation




(b) real world

Figure 3.3: Fitness development in simulation (a) and real world (b) generations for the three-interleaving condition. The vertical dotted lines indicate where the chain of generations in the simulator or real world have been interrupted by an interleaving.

| condition | $a$ | $b$ | $r^2$ | $p$ |
|---|---|---|---|---|
| **simulation** | | | | |
| no interleaving | 213.79 | 538.52 | .8443 | $< .0001$ |
| one interleaving | 167.75 | 5260.5 | .4870 | $< .0001$ |
| three interleaving | 161.08 | 6144.8 | .4699 | $< .0001$ |
| **real world** | | | | |
| no interleaving | -17.54 | 582.93 | .1289 | .0229 |
| one interleaving | -7.87 | 225.25 | .1547 | .0120 |
| three interleaving | 73.716 | -2123.7 | .4151 | $< .0001$ |

Table 3.1: Regression line coefficients ($a$ and $b$) of line $y = ax + b$, proportion of explained variance ($r^2$), and $p$-value for the regression line for each condition.

| Condition | Measurement 1 | | Measurement 2 | | |
|---|---|---|---|---|---|
| | Mean | Std. Deviation | Mean | Std. Deviation | N |
| no interleaving | 2040.60 | 595.77 | 1511.70 | 1335.96 | 10 |
| one interleaving | 2.00 | 6.33 | 4.60 | 10.224 | 10 |
| three interleaving | -189.80 | 281.82 | -331.20 | 581.73 | 10 |
| total | 617.60 | 1085.70 | 395.03 | 1150.37 | 30 |

Table 3.2: Mean and Standard Deviation of both measurements in each condition. See also Figure 3.4

## 3.2 Comparison of final fitness

A repeated measures multivariate analysis of variance (*repeated measures MANOVA*) over the ten best individuals of each condition showed that there is a significant difference between average fitness of the final generation in each condition ($F(2, 28) = 44.953$, $p < .001$, $\eta^2 = .769$). Summarized data can be found in Table 3.2. On further examination, pairwise comparisons showed that only the mean fitness of the no-interleaving condition differed significantly from the one-interleaving ($p < .001$) and three-interleaving conditions ($p < .001$). The difference of the mean fitness of the one interleaving and three interleaving condition was not significant ($p = .269$). See also Figure 3.4. The three groups of bars each represent a condition. The two bars each group consists of are the two measurements.

The difference in fitness between the two runs of each robot was not significant ($F(1, 28) = 2.508$, $p = .125$). This means that control structures in the final generation are able to reproduce their fitness scores (i.e. their *behavior*) over different runs.

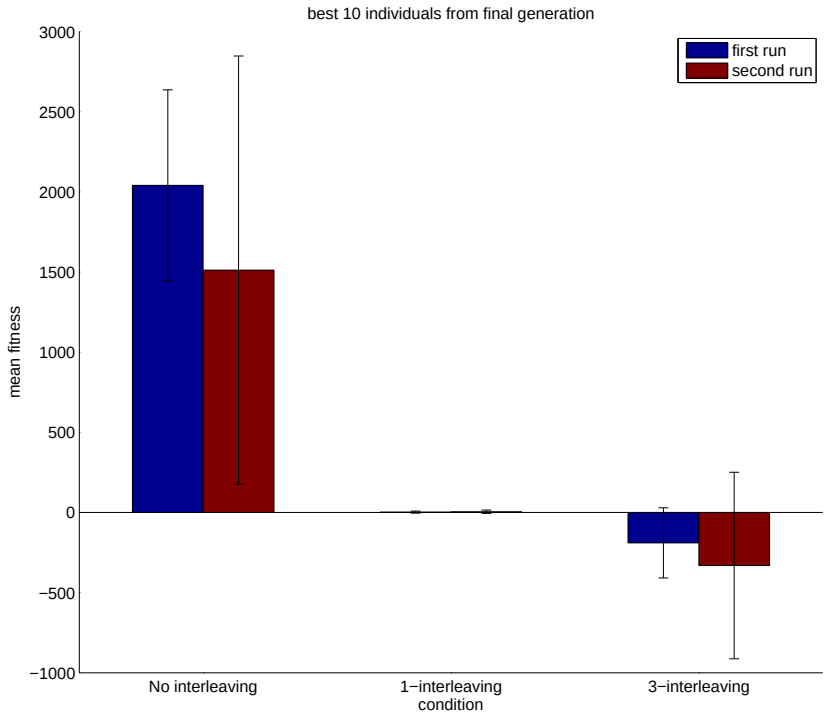



Figure 3.4: Difference in mean fitness of the best ten subjects of each condition. There is a significant difference between the no interleaving condition and the one and three interleaving conditions, but not between the one interleaving and three interleaving conditions (see text).

# Chapter 4

# Conclusion and discussion

In this chapter, the conclusions which can be drawn on basis of the results are discussed. Section 4.1 presents the conclusions, while the effect of real world exposure length is described in section 4.2. The performance difference of the control structures in simulation and the real world is discussed in secion 4.3.

## 4.1 Conclusions

Both the experiment in this thesis and the experiment of Goosen (2007) were only run once (see Section 4.5). All conclusions drawn from the results presented is tentative, but despite this, a number of interesting observations can be made.

After analysis of the mean fitness of the best 10 individuals of the final generation in each condition, the following can be concluded. Both interleaved conditions performed worse than the condition in which no interleaving was used. The no interleaving condition performs significantly better than both interleaving conditions. Fitness levels of the one interleaving and three interleaving condition do not significantly differ from each other. We can now use these results to evaluate the predictions we made of the possible outcomes of the experiment from Section 2.6.

The first and third predictions, which state that interleaving real world generations with simulation generations should cause an improvement in fitness, are not supported by the empirical evidence gathered in this study. Also, the second possibility, which states the effect Goosen (2007) measured might have solely to do with the real world exposure time in his interleaving condition, is not supported by the data. The fifth prediction fits the data pattern best, since the no-interleaving condition performs best, followed by the one-interleaving and third-interleaving conditions, albeit that this last difference is not significant. This means that there is an effect of real world exposure duration. This conclusion are discussed in depth in Sections 4.2 and 4.3.

Performance of control structures was bad compared to their fitness scores in simulation. Although fitness values in simulation and real world cannot be compared, as mentioned before, the difference is striking. Possible explanations are a bad simulator, or the effect of embodiment of control structures in general. This is discussed further in Section 4.4.

As already stated in Section 3.2, a control structure reached roughly the same fitness level every time it was tested. The evolutionary process used resulted in the selection of individuals

that produced reliable behavior.

## 4.2 Influence of interleaving

The results from this experiment did not confirm that the effect described by Goosen (2007) was caused by interleaving real world generations with simulator generations. The present experiment did produce a significant difference in the average fitness levels of the final generations. This, in turn could have been caused by different continuous real world exposure lengths (see Section 4.3).

The results in this study indicate that bridging the reality gap is not as straightforward as it may look at first glance. The large drop in performance which occurs after the control structures are placed in the real world can be contributed partially to the different scales in fitness. However, it may also be an indication that the current simulator is not well suited for the evolution of control structures. in the way presented in this study as well as Goosen (2007). Perhaps the role of embodiment and embeddedness of a control structure is underestimated in general. Simulators will never be able to simulate every unique detail of a given robot. The real robot, with its unique sensors and motors, is always its best own representation.

## 4.3 Effect of real world exposure duration

When looking closer to the real world evolution of all three conditions, only the no-interleaving condition shows a sustained increase in both maximum and mean fitness values. The one-interleaving condition does show some improvement in the maximum fitness, but this improvement disappears after a few generations. For both blocks of evolution in the real world, the average fitness level stays close to zero. The average fitness of the three-interleaving condition does increase, especially in the last three blocks of real world evolution, but the maximum fitness value stays at the level it begins at.

Since the no-interleaving condition has twelve uninterrupted generations of evolution in the real world, the high fitness levels of the final generation of this condition might be caused by the fact that the evolutionary process is able to work better when it is not interrupted by simulation generations. This indicates that the simulation is not compatible with the real world, and moving control structures between the simulator and the real world might do the evolutionary process more harm than good.

## 4.4 Performance in real world and simulation

As can be seen in Figures 3.1, 3.2 and 3.3, there is a drop in performance every time a generation is placed in the real world. The population of control structures is able to reach an average fitness score of about 6000-7000, but initial performance in the real world is around 0 at best. The average fitness score of the three-interleaving condition is even below zero. Drops in performance can also be observed when the control structures are placed back in the simulator, but these are only marginal.

The severe drop in performance when the control structures where moved from the simulation to the real world was not observed by Goosen (2007). In his experiment, control structures

were able to use the experience that was gained in the simulation in the real world, especially in the interleaving condition. Perhaps this effect caused by different lighting conditions between this experiment and that of Goosen (2007). It is clear that even the slightest change to light conditions can have a profound effect on the values returned by the light sensors of the robot.

There is also a difference in maximum and mean fitness between the no-interleaving and one-interleaving condition (which are both around 0) and the three-interleaving condition (which is below zero) at the start of the first block of real world generations. This can be caused by different local maxima being reached in the different conditions.

Also, there is a difference in the way conditions react to the real world. For instance, generation 20 of the one-interleaving condition and generation 10 of the three interleaving condition both have an average fitness around 12000 just before they are placed in the real world. In the next generation, which takes place in the real world for both conditions, the average in the one-interleaving average is 0, while the average of the three-interleaving condition is -5000. Such differences also suggest that different maxima are reached in each condition due to the stochasticity of the evolutionary algorithm. When more runs of this experiment are made, this effect will probably be averaged out.

In all, even though it is not possible to compare the fitness values of generations in simulation and the real world, the difficulty of reaching an acceptable performance in the real world after a number of evolved generations in simulation make claims made in this experiment and that of Goosen dubious. It is clear that the Lego Mindstorms Simulator is not well suited for experiments in which the simulator prepares robot behavior for the real world. This problem might be caused by differences in the way the control unit is represented simulator and the way it really works. Also, different reported light values between simulated and real world light sensors possibly play a part. Further investigation is needed to validate this claim, however.

## 4.5 Validity

Because of the time limitations concerned with a Bachelor thesis, decisions were made that affected the validity of the results of this thesis. The main reason for this is that only one single run was performed, and each control structure could only be tested once or twice in each generation. As stated in Section 2.5, the experiments should ideally be performed 16 to 20 times so it is possible to do a T-test on the results. Also, every control structure should be tested multiple times.

Repeating the experiment in its current form is very time consuming, but a number of improvements can be made to reduce the time needed for a single run. First, the speed of the simulator can be improved by eliminating graphic output. Second, more than one session of the real world trials can be performed in parallel by different groups. Alternatively, an entirely new paradigm could be chosen that is simpler and faster to run than the task, simulator and robot presently used.

To see whether the drop in performance is persistent in every generation, or if the results in this study are based on bad luck, future research could be done by placing the best individual from every simulation generation in the real world, like done in the research by Nolfi, Miglino, and Parisi (1994).

# Chapter 5

# Future research

This section describes some regions of interest where future projects in line with this thesis could focus on.

## 5.1 Multi-objective optimization

The task in this experiment has two parts. First, the robot needs to maximize the distance traveled in the environment. Second, the robot needs to minimize the number of bumps. That means the task used is *multiobjective*. The fitness function, however, only consists of a single value for its total performance. This value is based on the traveled distance, from which an arbitrary penalty is subtracted for every collision registered by the bumper. It is therefore not possible to see the difference in a robot which goes nowhere and a robot which only goes straight and gets stuck in a wall. While the first robot is arguably better in avoiding walls, the other robot clearly outperforms the first robot in the traveled distance. Such a distinction is important, especially in the early stages of the evolutionary process where the majority of the robots go nowhere and early robots get stuck in the walls easily.

Special evolutionary algorithms have been developed for multiobjective tasks. These algorithms are usually based upon the use of a Pareto optimal front to order the multiple fitness values for each individual, a method proposed by Goldberg (1989). With a Pareto front, solutions which excel in one or more areas get selected, even though their performance is not great in other areas.

## 5.2 Crossover operator for ANNs

The exclusion of the crossover operator in the evolutionary process kept the setup for this and its preceding experiment (Goosen, 2007) simple. Also, the classic crossover operator is not beneficial when evolving ANNs due to the permutation problem (Yao, 1998, 1999). However, it is possible to implement one if the right heuristics are used. Such implementations have been made (e.g., Spronck, Sprinkhuizen-Kuyper, and Postma (2001)). It would be an interesting project to compare different crossover strategies for ANNs and see how the crossover operator can be best implemented for control structures that have to perform similar tasks as presented in this study.

# References


Floreano, D., & Mondada, F. (1994). Automatic creation of an autonomous agent: Genetic evolution of a neural-network driven robot. *From Animals to Animats*, *3*, 421-430.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization and machine learning.* Addison-Wesley.

Goosen, A. E. A. (2007). *Evolving in advance: Interleaving simulated and physical environments in robot evolution.* Unpublished Bachelor's Thesis, Radboud University Nijmegen.

Goosen, A. E. A., van den Brule, R., Janssen, J. H., & Haselager, W. F. G. (2007). Interleaving simulated and physical environments improves evolution of robot control structures. *Proceedings of the 19th Belgian-Dutch Conference on Artificial Intelligence*, 135-42.

Holland, J. H. (1975). *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence.* The MIT Press.

Künsting, B. (2004). *Konzeption und prototypenhafte implementation einer lego-mindstorms simulationsumgebung -lejos-emulation.* Unpublished master's thesis, Universität Paderborn.

Miglino, O., Nafasi, K., & Taylor, C. E. (1994). Selection for wandering behavior in a small robot. *Artificial Life*, *2*, 101-116.

Mitchell, M. (1996). *An introduction to genetic algorithms.* The MIT Press.

Nolfi, S., & Floreano, D. (2000). *Evolutionary robotics: The biology, intelligence, and technology of self-organizing machines.* The MIT Press.

Nolfi, S., Floreano, D., Miglino, O., & Mondada, F. (1994). How to evolve autonomous robots: Different approaches in evolutionary robotics. *Artificial Life*, *IV*, 190-197.

Nolfi, S., Miglino, O., & Parisi, D. (1994). Phenotypic plasticity in evolving neural networks. *From Perception to Action Conference, 1994., Proceedings*, 146-157.

Spronck, P. H. M., Sprinkhuizen-Kuyper, I. G., & Postma, E. O. (2001). Evolutionary learning of a neural robot controller. *Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation*, 9-11.

Sträter, W. (2004). *Konzeption und prototypenhafte implementation einer lego-mindstorms simulationsumgebung -simulationsmaschine.* Unpublished master's thesis, Universität Paderborn.

Walker, J., Garrett, S., & Wildon, M. (2003). Evolving controllers for real robots: A survey of the literature. *Adaptive Behavior*, *11*, 173-203.

Yao, X. (1998). Towards designing artificial neural networks by evolution. *Applied Mathematics and Computation*, *91*, 83-90.

Yao, X. (1999). Evolving artificial neural networks. *Proceedings of the IEEE*, *87*, 1423-1447.