Reality strikes back:

The problems with a real-life implementation of Evolutionary Robotics

Bachelor Thesis
By: Maarten-Jan van Gool
Affiliation: Department of Artificial Intelligence, Radboud University Nijmegen
Student Number: s0513393
March 9, 2009

Contents

Abstract

Evolutionary robotics can be done in two ways; in simulation and in real-life. Earlier studies have showed that interleaving does not show very promising results. In this paper I did a predator-prey experiment completely in real life and I compared it with the results of simulation. The main research questions are: is it possible to get results in real life experiments that show progress? And is this progress comparable with progress in simulation? What are the main difficulties with implementing a real life experiment?

Keywords: Evolutionary Robotics, Artificial evolution, Predator-Prey, simulation, real-life evolution

Acknowledgements

I would like to thank Pim Haselager and Ida Sprinkhuizen-Kuyper for their support, guidance and expertise. Without their help, I would never have reached this point. I would also like to thank Jules Ellis, for helping me with the statistical analysis.

1 Introduction

In this chapter I will explain my motivations for the experiment I ran. I will start by introducing the field of evolutionary robotics. After that I will zoom in on co-evolution and predator-prey evolution; the main subject of this thesis. I will discuss my research questions; and I will also explain how results in this part of the field of Evolutionary Robotics are obtained.

1.1 Evolutionary Robotics

The solution to a (computational) problem can be found in a lot of ways. The simplest way to find an answer is to perform a random search. Of course, if the solution space is very large, it could take forever to find a suitable solution. In order to solve this problem, a lot of heuristics were created. A heuristic is a way to speed up the search process by preferring possible solutions that are 'generally' more likely to work.
However; in order to use most heuristics a lot of domain knowledge is needed to make it work; else the risk is too great to get stuck in a local optimum.

Evolutionary algorithms and evolutionary robotics are based on Darwinian evolution: 'survival of the fittest'. Holland (1975) proposed at first an evolutionary algorithm. All that is needed is a starting population, a fitness function and an evolutionary algorithm. The evolutionary algorithm mutates and/or crossbreeds the individuals the fitness function finds fittest in order to generate a new population. After that the program can repeat itself until a suitable solution is found.

The advantage of the evolutionary algorithm is that it works on almost every problem. This is because there is not much domain knowledge needed in order for it to work: it is not necessary to know what the solution should look like. Most of the times an evolutionary algorithm has more flexibility to avoid local optima if compared to other heuristics.

There are multiple reasons to use evolutionary algorithms in robotics. One reason could be that the control mechanism, a neural network, functions as a 'black-box'; thus it is not known what the solution would look like. Often the solutions are completely different of what is expected.

Robotics are often used as a simulation of real-world (inter)actions between animals. In order to simulate evolution of animals evolutionary algorithms could be added. Interaction between different populations while they are evolving is called co-evolution.

1.2 Co-evolution: predator-prey

Co-evolution is used to train different populations in the same experiment. There are multiple ways of co-evolving robots: let them work together or let them compete. In this thesis I will only discuss the latter case. With competing robots the goal is to stimulate one population by improvement of the other. In essence the goal is to start an 'arms race' (Nolfi & Floreano, 1998), like they occur in real-life situations (Dawkins & Krebs, 1979) (also see the red queen effect (Van Valen, 1973)). One example of competing robots is the predator-prey relation. In that case there are two populations: one population hunts the other.

However; improvement because of co-evolution is not always guaranteed. It is for example possible that the predators get almighty. If this happens; evolution stops: prey has no chance of improvement, and predator does not have to improve.

Another problem is called cycling. For example: the predator has tactic A, while the prey has the less successful tactic X. In order to beat tactic A, the prey 'comes up' with tactic Y. In order to beat the prey again, the predator switches to tactic B. And then the prey switches again to tactic X, which is not effective against A, but is effective against B. This will make the predator switch to A. Cycling has begun.

1.3 Interleaving

Because of the fact that evolving real robots takes a lot of time most of the research is done in simulation. Another possibility is to evolve the robots partially in simulation, and partially in reality. This is called interleaving. Goosen (2007) and Van den Brule (2008) ran some experiments to examine if this approach works.

Unfortunately, they both concluded that interleaving is not a good solution for multiple reasons. First of all, resetting the trial in the real-life part of the experiment was done by hand, wasting a lot of time of the experimenter. Secondly, the difference between reality and simulation appeared to be too big to overcome (this is called the reality gap). The improvement obtained in the simulation did not result in improvement in real-life. The first problem I solved by automating the process of resetting. The second problem I avoided by conducting my experiment completely in real life; I will not use simulation.

1.4 Obtaining results

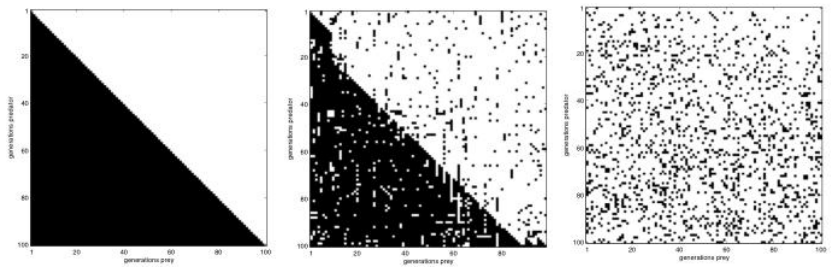
Because of the risk of cycling, it is difficult to evaluate the obtained results. One way of evaluation is to use master tournaments. The champion of each generation of predators is competing against all champions of every generation of preys. If later generations beat the earlier ones; there is no cycling taking place, and you obtain good results; the robots are improving. Figure 1 (Van den Braak, 2005) shows what the results would be like in different situations. The pictures in figure 1 show what the results would be like when a population of predators competes against a population of prey for 100 generations.

In the ideal situation, the predator wins when it is of a higher generation than the prey. This should also work the other way around. In this case the results would be like Figure 1(a). Unfortunately, results are most of the time far from ideal. It is more realistic to get results like the ones in Figure 1(b). Figure 1(c) shows possible results that are obviously not preferred.

It can be argued that the name for Figure 1(c) should be 'random behaviour' or 'no improvement' instead of cycling because it is not clear from the picture that old tactics are reused. However; since the only thing important is the quality of the results; and because this Bachelor thesis limits me because of time restrictions to investigate this further; I would like to point to my source of the pictures, Van den Braak (2005), for further information.

1.5 Simulation of robotics

Like I said in paragraph 1.3, most evolutionary robotic experiments are done in simulation. Van den Braak (2005) wrote her master thesis on predator-prey simulation. In this paragraph I will discuss one of the experiments she did. In



(a) Ideal (b) Progress (c) Cycling

Figure 1: Here you can see the ideal situation, co-evolutionary progress and cycling. On the $x-$axis are the generations of the prey set out (from 0 to 100, from left to right). On the $y-$axis are the generations of the predator set out (from 0 to 100, from top to bottom). A black dot means the predator has won; a white dot means the prey has won (Van den Braak, 2005).

the first part I will discuss her methods. In the second part I will talk about her results and her conclusion.

1.5.1 Methods

Van den Braak used the Evorobot software (Nolfi, 2000) to run her simulations. The simulation software simulates 2 Khepera robots. Both Khepera robots were equipped with 8 infrared sensors and 5 photoreceptors.The robots were placed in an empty environment, surrounded by walls. The robots had 2 motors, which were controlled by a neural network with 30 synapses. Each weight was encoded in 8 bit. The genotype was encoded in a bitstring that had the length of $30 * 8$ bits. The prey was able to go twice as fast as the predator.

1.5.2 Results

Van den Braak ran her simulation for 100 generations; and she did this 10 times. She also conducted master tournaments of her results, and the best results she obtained are shown in figure 2.

1.5.3 Conclusion

Van den Braak concluded that her results were not as good as the results of (Nolfi & Floreano, 2000). Because of the speed advantage of the prey and the fact that the prey can see the predator coming, the predator had a considerable

Figure 2: The results of van den Braak (2005). On the $x-$axis are the generations of the prey set out (from 0 to 100, from left to right). On the $y-$axis are the generations of the predator set out (from 0 to 100, from top to bottom). A black dot means the predator has won; a white dot means the prey has won.

disadvantage. However, in my opinion it is still quite clear from the figure that higher generations are having a better chance at beating the lower generations. So there is at least some progress. The statistical analysis Van den Braak performed also supports this view.

1.6 My experiment

In this bachelor thesis I will conduct an experiment with 2 real lego Mindstorms robots. Each robot has its own task: one of them will be the predator; and the other will be the prey. My goals are the following:

- Figure out the capacity of the Lego Mindstorms NXT to work with evolutionary algorithms and neural networks.
- To find out if introducing auto-resetting of a trial improves the experimental conditions for the experimenter, and reduces the length of the experiment in relation to the results of Goosen (2007) and Van den Brule (2008).

And my main research question:

- I want to find out if predator-prey co-evolution works in a real-life setup,

and how well these results are compared to the results of Van den Braak (see section 1.5).

The exact methods I did use are discussed in the next section. I will also elaborate on my main research question there. The results I obtained can be found in section 3; which will be discussed in the conclusion (section 4). Suggestions for further research are done in section 5.

2 Method

In this section I will discuss the methods I used; I will first discuss the setup of the arena and the robots. After that I will state my expectations.

2.1 Arena

The arena has a white floor which is 140 centimeters in length, and 140 centimeters in width. It is surrounded by white walls. There won't be any obstacles in the way.

2.2 Hardware

I am using Lego Mindstorms NXT for this experiment. In this package, the brick is the central part of each robot. It is possible to connect up to 3 motors and 4 sensor units. Each robot will get 2 motors, to drive around with, and 4 sensors: 2 sonars, and 2 light-sensors. The sonars are used to measure distance, while the light-sensors are used to measure light intensity. Because each sensor-type is placed on the left and on the right side of the robot, it should be able to tell the difference between left and right. To make sure the light sensors are effective, I put black paper around each robot. One of the robots is shown in Figure 3. Both the robots will have the same morphology and software. The only thing different between the predator and prey is the fitness function. I chose not to give the prey a speed advantage, because Van den Braak found that favored the prey too much. This would disturb the evolutionary progress. Although because of other reasons my results may be less compared to the results of Van den Braak, I expect this specific change will improve the results. In Table 1 the hardware parameters are summarized.

Parameter:	Value:	Source:
Arena-size	140x140	Own decision
Sensor	2 Sonars	Own decision
	2 light-sensors	Own decision
Speed-advantage prey	0	Van den Braak
Speed-advantage predator	0	Van den Braak

Table 1: Hardware parameters summarized.

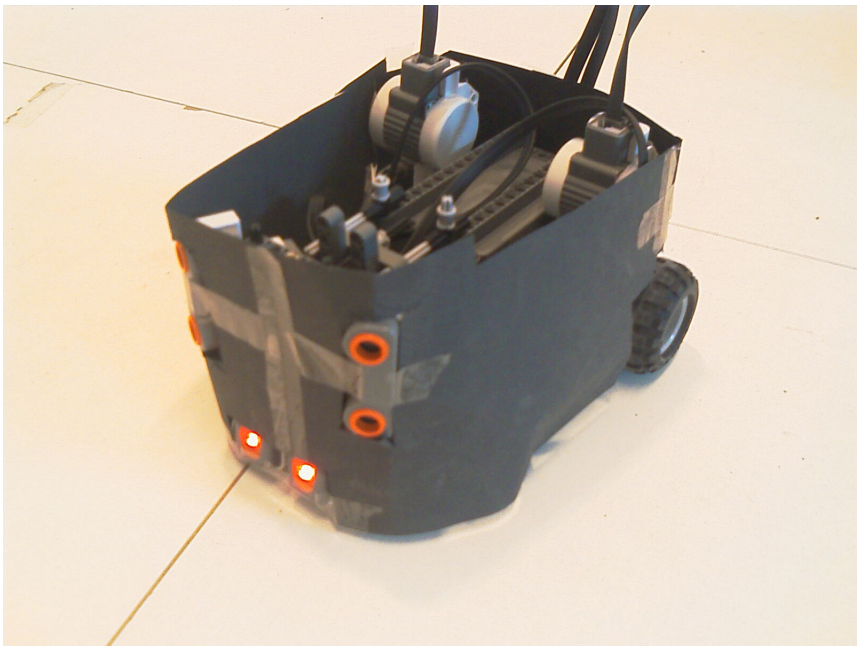


Figure 3: One of the bots

2.3 Software

The sensors will give their output to a neural network (see figure 4), that will control the motor units. The neural network will have 6 input-units (2 sonar readings, 2 light-sensor readings and the 2 previous motor readings) and 2 output units. Between the input-layer and the output-layer there will be a hidden-layer, together with the context layer (Elman, 1993). The context layer contains the activations of the hidden layer of the previous time step, and functions as a memory. These layers will each contain 6 nodes. This will give a neural network with in total $6 * 6 + 6^2 + 6 * 2 = 84$ weights. Every weight will be encoded in

Parameter:	Value:	Source:
Recurrency	yes	Own decision
Number of Synapses	84	Own decision
Weight-size	8 bit	Van den Braak

Table 2: Parameters of the neural network summarized.

8 bits (the bits were encoded to an integer, positive or negative was decided by the first bit, to compute the motor output). The genotype will thus contain $84 * 8 = 672$ bits. My neural network is bigger then the network used by Van den Braak and has a recurrence the network of Van den Braak has not. I did this because I found out that if I made the network smaller, or when I removed the context-layer, the robots would perform the same action every time they were in the same situation. Obviously, this is not desirable behavior; for a good fitness some flexibility is required.

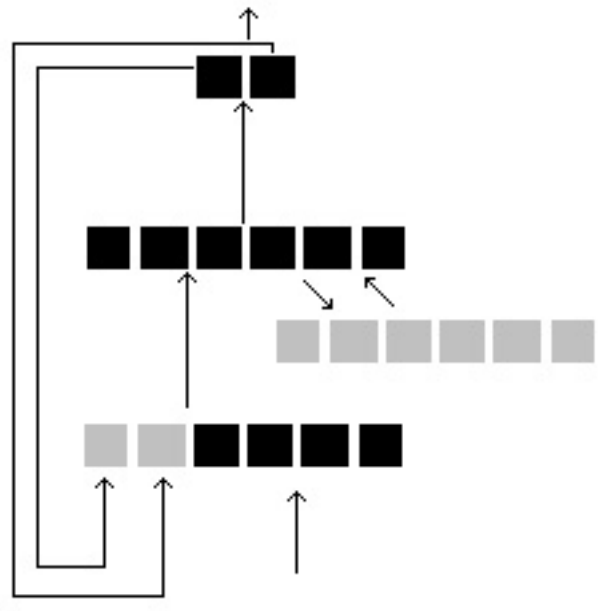


Figure 4: The Neural Network (Grey nodes are recurrent)

A trial of 1 individual robot will take 400 time-steps. Every step will take

Parameter:	Value:	Source:
Number of Generations	40	Own decision
Population Size	20	Van den Brule
Number of offspring	2	Van den Brule
Mutation rate	0.5%	Own decision

Table 3: Parameters of the genetic algorithm summarized.

approximately 200 milliseconds. Every step the network receives new input and the motors get new instructions.

Goosen (2007) and Van den Brule (2008) have inspired me to choose 2 populations with 20 individuals each (100, like Van den Braak would be a practical impossibility because of time-restrictions). I chose a mutation rate of 0.005. I ran the experiment for 40 generations. In every generation, the best 10 predators and the best 10 prey will have 2 'children' (the same setup was used by Van den Brule (2008)). I chose to follow Van den Brule here, because I also did this with the population size. Only mutation will take place; cross-over is not used in this experiment. I chose the mutation rate to be low, because I found in earlier 'trial-and-error' that a higher mutation grade leaded to behavior totally different from parental behavior. These parameters are summarized in Table 3.

2.4 Fitness function and the initiation of the algorithm

Every new run, a predator was randomly paired (under the condition that they were not selected earlier) with a prey. The fitness of a predator is determined by the number of time-steps it took him to capture the prey. If it is not able to capture it within the time limit, it will get a fitness of 400 (the maximum number of time-steps). The best predator in a generation can be determined by:

$$\min_{i=1}^{20}(t_i)$$

The fitness of a prey is determined by the number of time-steps it was able to avoid the predator. If it was able to avoid the predator the entire run; it also got a fitness of 400. The best prey in a generation can be determined by:

$$\max_{i=1}^{20}(t_i)$$

Please keep in mind the fitness of the prey is the opposite of the fitness of the predator. A high score for example is not good for the predator, while being very well for the prey.

The evolutionary algorithm is started with a population with random genotypes. Every value in the genotypes is chosen by a random value generator.

2.5 Expectations

There are multiple outcomes of this experiment possible:

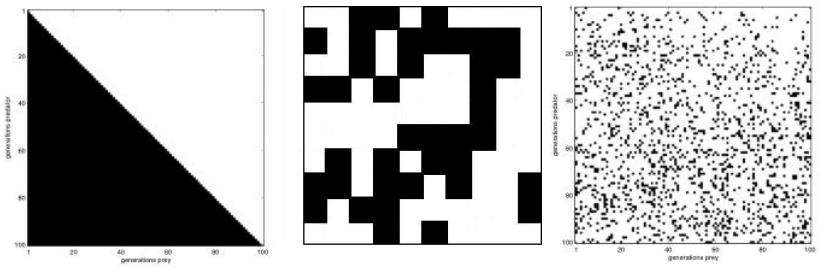
1. The real-world experiment works better than the simulation.
2. The real-world experiment is just as good as the simulation.
3. The real-world experiment is not as good as the simulation; but still progress is visible.
4. The real-world experiment does not work at all; and we don't get close to the results obtained by Van den Braak. This can happen in three ways:
   (a) The hunter always wins.
   (b) The prey always wins.
   (c) Cycling takes over.

I do not expect it will be the first or the second possibility. Because simulation is simpler than the real world and the fact that in simulation far more runs can be done, it does not seem likely real-life will beat simulation or even get the same results.

It is far more likely that option 3 or 4 will be the result. If the results are like option 3, real-world evolutionary robotics has a chance of succeeding; even though interleaving does not yield promising results. When the results won't be satisfiable; the experiment should be run in different ways. If option 4(a) or 4(b) takes place; the population that always loses should get an advantage, and the experiment could be run again. If cycling (or random behavior) takes place, it will be more problematic to make things work.

3 Results

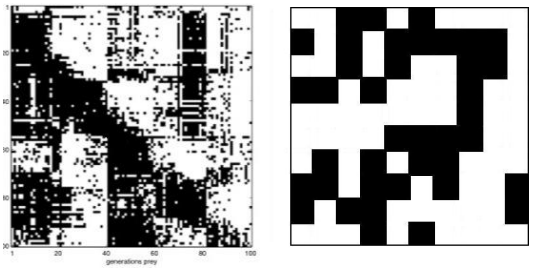
To find out if my robots improved I conducted a master-tournament. Because of time restrictions; I only evaluated every fourth generation. The visual results of the master-tournament can be found in figure 5(b). As can be seen; the results do not look very promising. If compared with the ideal situation and compared with the cycling example, my results seem to have more in common with the cycling example.



(a) Ideal (b) My results (c) Cycling

Figure 5: My results versus Ideal and cycling (predator from top to bottom; prey from left to right). Remember; my results are compared to example pictures from Van den Braak. I evaluated 10 generations, while she evaluated 100.

If compared with the results obtained by Van den Braak (Figure 6), it does not look similar at all. Looking at the pictures the difference between simulation and real-life seems very big. In the next chapter I will conduct a statistical analysis, to further analyze the results.



(a) Results of Braak (b) My results

Figure 6: My results versus the results of van den Braak. (predator from top to bottom; prey from left to right). Remember; my results are compared to those from Van den Braak. I evaluated 10 generations, while she evaluated 100.

3.1 Statistical Analysis

Looking at the figures the results do not seem very promising. However, since looks can be deceiving, I performed an isotonic trend analysis (Ellis, 2005) to find out if statistics support my earlier claims. The isotonic trend analysis is used to find out if there is a significant increase in performance over the generations. In this test an expected increase is postulated from the data. There are two questions to be answered:

- Do the results from the master tournament differ significantly from the expected increase? If the expected increase does not differ significantly from the master tournament results; the expected increase can be used as a representation for these results.
- Is the expected increase significant?

In order to find out if the performance decreased, I ran the test the second time with reversed master tournament results (the tenth generation I tested became the first, the ninth the second, et cetera). If the test results do not show a significant increase, there is no proof there is a decrease in performance. The p values of the difference between the expected and actual results (deviation) and the improvement (regression) can be found in Table 4 and Table 5.

In order to clarify; what is wanted is an expected result that does not differ significantly from the actual master tournament results; while the expected result shows a significant increase in performance in the test. To make it short, in the test deviation has to be low (with a high p value) while regression has to be high (with a low p value). Because we do not want to see a worsening population; the results should be the other way around for these tests (Table 5).

The results of improvement are most interesting. There is no evidence that the predator is improving. However; because of the small n, it is probable my test was too small to find a difference, even when it was there. For the prey a marginal significant difference in regression can be found ($p = 0.060$). It seems the prey improves in the last part of its evolution (also see Figure 8 in appendix B).

As can be seen in Table 5, there is no significance to be found in a decrease of performance. This could also be the case because of my small n, but there is

at least no proof of a decrease; which is a good thing.

	Predator		Prey	
	p-value	R^2	p-value	R^2
Deviation	0.213	10.9%	0.965	2.5%
Regression	0.537	0.3%	0.060	3.8%

Table 4: P values and R^2 of the improvement.

	Predator		Prey	
	p-value:	R^2	p-value:	R^2
Deviation	0.194	8.7%	0.663	6.1%
Regression	0.461	2.5%	0.689	0.2%

Table 5: P values and R^2 of the worsening.

I would further like to point out that the R^2 values for the prey point more to an increase of performance then to a decrease. In the improvement test the R^2 for deviation is relatively small; while R^2-regression is relatively big, while it is the other way around for the worsening test. This is not the case for the predator. Here the R^2-values point towards a decreasing performance.

4 Conclusion and Discussion

From the results we can conclude that the results from the test are promising, but not convincing. The difference between my results and the results obtained by Van den Braak seem very large; visually as well as in the statistical analysis.

However; although the prey results show improvement, I still believe that this experiment has some evidence it is at best very difficult to use evolutionary robotics in real life. I believe this for multiple reasons I will discuss in the upcoming paragraphs.

4.1 Hardware problems

The first problem I found would be that the hardware is not good enough to run evolutionary robotic experiments. In my case; Lego Mindstorm-robots are

not built for these kind of experiments. The sensors (light sensor and sonar) of the robots are very limited; this especially goes for the light-sensors (they have a very limited range). Furthermore, the bluetooth connection I used for communication between robots and computer, (more on this in the appendix) was very unstable at times; which made communication very difficult and time-consuming.
Another problem I encountered was that the batteries of the robots are depleted in 3 hours. This still made it impossible to run trials for a long time without supervisement.

4.2 Time problems

The battery problem and the bluetooth problem are related to the second reason I believe that it is very hard to implement a real-life version of evolutionary robotics. There is a limited time available. It takes 90 seconds to run 1 trial. In order to run 1 generation (of 20) would take half an hour ($\frac{90*20}{60} = 30$). And to run 40 generations; it took me 20 to 25 hours. Imagine how long it would take to run 100 generations with a population-size of 100 and that 10 times like Van den Braak did. This is simply impossible.

The time problem is especially problematic looking at the results of my experiment. According to the results; the prey starts improving in it's final generations. If given more time; the prey could have pressured the predator to improve as well; starting the arms race. Although this hypothesis is speculative, because of the stated time problem, it seems impossible to test.

In spite of the above problems; some improvements are made; which made the working of the experiment a bit more comfortable. While Van den Brule had to 'reset' the track manually after every run; this was not necessary in my set-up (my robots 'semi-randomly' repositioned themselves after a trial). This surely saved a lot of time and energy.

4.3 The reality Gap

The robots have a more difficult environment in real life; but they have less generations to evolve in it. The only conclusion we can draw from that is that

the robots will do worse in real life than in simulation (assuming that the simulation is simplified in a way profitable for the robots). And in my case; only the prey shows a marginal improvement.

4.4 Validity

Because of the fact that I ran only one experiment; I do have to question the validity of my results. It may very well be that I just had bad luck. There is still a lot of research that can be done to the parameter settings of evolutionary algorithms; it may very well be I chose very bad parameters. As I already pointed out; my population is far too small to conclude real-life co-evolution is impossible.

However; I still think my general conclusion stands. I do not think the time and hardware problems I encountered were the result of bad luck or wrong parameter settings. The reality-gap is very real. And in order to fill this gap more is needed then only fine parameter settings and perfect hardware. In order to find really meaningful results; these problems have to be solved.

5 Suggestions for further Research

In this section I will do some suggestions for further research.

5.1 Search for optimal parameters

As I already said in my conclusion; my parameter-setting is quite arbitrary. There is still a lot of research that can be done in order to find optimal parameter settings; up until now there are no guidelines to do this. I would like to point towards the research of Eiben and Schut (2008), who are already working on just this.

5.2 Solving time limitations

In order to solve the time limitations I encountered; a lot of work has to be done on simulation software. If simulations simulate real life experiments more accurate; the reality gap may become smaller. If it is small enough; results of

simulation may successfully be implemented in real-life. This will not only solve the time-problems there are; but also will overcome the reality gap.

5.3 Hardware setup

Probably better results can be obtained using better hardware then I did. Especially because of the time limitations faced using real robots it is important that the robots can make use of accurate sensors; a long lasting battery and a stable bluetooth connection.

5.4 Decreasing the search space

While it is very difficult to gather large amounts of data; it may be necessary to decrease the search space. Easier problems would require less data to show significant results. It would also be more difficult to generalize; but since it is more difficult to say anything about non-significant results; this does not seem like a big obstacle.

References

Dawkins, R., & Krebs, J. (1979). Arms races between and within species. *Proceedings of the Royal Society of London B, 205*, 489-511.

Eiben, A., & Schut, M. (2008). New ways to calibrate evolutionary algorithms. *Advances in Metaheuristics for Hard Optimization, Natural Computing Series, Springer*, 153-177.

Ellis, J. (2005). Testing the hypotheses about the order of means in anova. *Unpublished Manuscript, Nijmegen Institute for Congnition and Information, Nijmegen, The Netherlands.*

Elman, J. (1993). Learning and development in neural networks: the importance of starting small. *Cognition*, *48*, 71-99.

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

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

Nolfi, S. (2000). *Evorobot 1.1 user manual (technical report).* Rome, Italy: Institute of psychology.

Nolfi, S., & Floreano, F. (1998). Coevolving predator and prey robots: Do arms races arise in artificial evolution? *Artificial Life.*

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

Van den Braak, S. (2005). Co-evolutionary robotics in simulation. *Unpublished Master Thesis, Radboud University Nijmegen.*

Van den Brule, R. (2008). The search for a bridge over the reality gap effects of interleaving interval duration and quantity. *Unpublished Bachelor Thesis, Radboud University Nijmegen.*

Van Valen, L. (1973). A new evolutionary law. *Evolutionary Theory 1*, 1-30.

Appendix A: Programming on PC and Mindstorms

In this appendix, I will explain some parts of the process I went through to program the robots and the pc.

Programming on the pc

I used the pc as the master, to control both the hunter and the prey. The programming language I used was c++, and the compiler I used was Dev-CPP. The pc was responsible for generating and mutating the synapses of the network, keeping the score and saving the data. With every new run the pc would send a new genotype to both the predator and the prey.

The library I used for bluetooth communication was the NXT Bluetooth c++ library from Anders ($http://www.norgesgade14.dk/bluetoothlibrary.php$). This library made it possible to send (and receive) strings to the robots. After a lot of experimenting with this library (documentation on it was almost non-existent) I was able to send the network relatively fast to the robots (would take up to 10 seconds (in comparison: in my first attempts it would take up to 2 minutes)). After the network was sent, the program would wait for the score of the robots (which the predator sent; the prey has precisely the opposite score), and after that a new run would start. When a complete generation was finished, the old generation was saved; and a new generation was generated.

Programming on the robots

The software on the robots contained the program to run the neural network. The programming language I used was Not eXactly C (NXC) and the compiler I used was Bricx Command Center (BricxCC). The robots ran the enhanced NBC/NXC firmware (version 1.05). The individual robots were responsible for executing one run. They both started after they received their genotype; and finished when either the predator captured the prey; or the time ran out. At that point the predator would send the score back to the pc (the score for the predator was calculated from that), the robots would perform some semi-random movement to reset the stage and the algorithm would restart.

Programming schematics

In Figure 7 you can see how the program works.

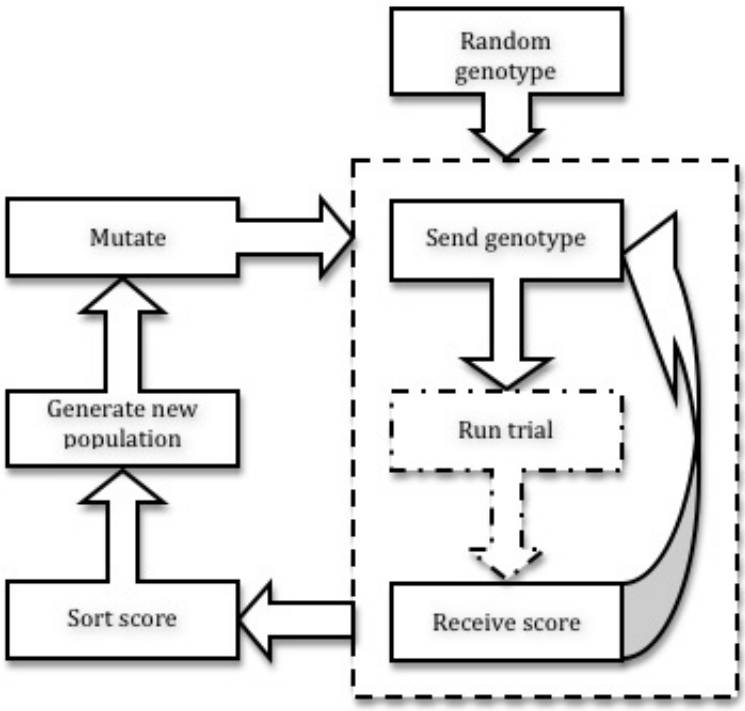
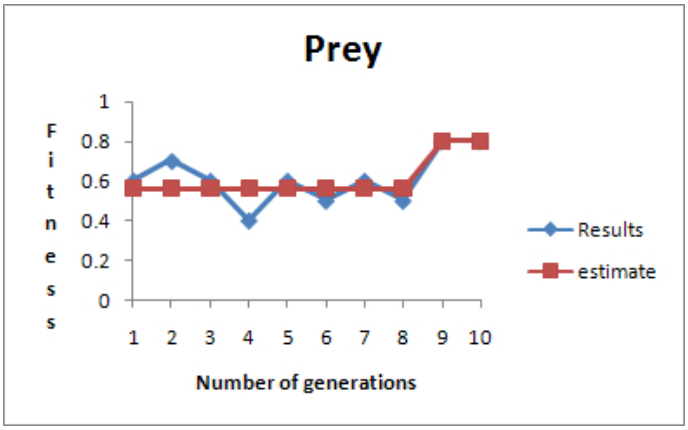


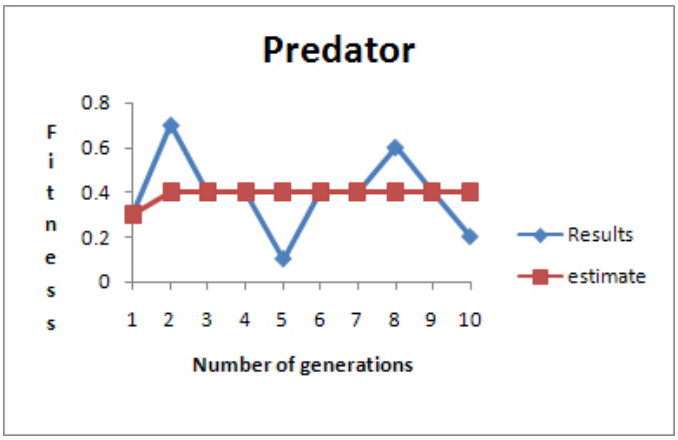
Figure 7: The programming schematics. In straight lines all the processes that run on the pc; only 'Run trial' runs on the robots. Everything inside the dotted line are processes of an individual; everything outside the dotted line are processes of a generation.

Appendix B: Graphs of the results

In these graphs (figure 8) the results of the predator and prey can be seen, as well as the estimated improvement.



(a) Prey



(b) Predator

Figure 8: Actual and estimated results for predator and prey. Note that I evaluated every fourth generation; in order to find the generation evaluated; the number of generations has to be multiplied with four.