Dynamical Properties of the Fitness Landscape of a GP Controlled Random Morphology Robot

Peter Dittrich
Dept. of Computer Science
University of Dortmund
Germany

Andre Skusa
Dept. of Computer Science
University of Dortmund
Germany

Wolfgang Banzhaf
Dept. of Computer Science
University of Dortmund
Germany

Wolfgang Kantschik
Dept. of Computer Science
University of Dortmund
Germany

Email: `<lastname>@LS11.cs.uni-dortmund.de`
WWW: `http:ls11-www.cs.uni-dortmund.de`

Abstract

The aim of this contribution is: (1) to present an easy to maintain robot hardware platform which allows on-line evolutionary experiments and demonstrations; (2) to introduce a simple method to measure dynamical characteristics of the time-dependent fitness landscape by using reference individuals; (3) to demonstrate dynamical properties of the fitness landscape based on fitness measurements of reference individuals. The implication of the observations for the design of on-line EAs in time-dependent fitness landscapes are discussed.

Keywords: genetic programming, evolutionary robotics, on-line evolution, dynamical fitness landscape, reference fitness

1 INTRODUCTION

In a conventional evolutionary algorithm (EA) [1] it is assumed that *one* fitness value is assigned to each individual[1]. In a typical implementation the fitness is evaluated, when the individual is created and stored together with the individual in the population data structure. In this case the fitness landscape[2] can be expressed as a mathematical function. If the fitness function is non-deterministic, the fitness of an individual can be described by a probability distribution. If this probability distribution changes over time we call the fitness landscape **dynamic**.

Dynamic fitness landscapes are typical phenomena encountered in the field of evolutionary robotics especially in on-line evolutionary experiments on real robots [5, 10, 12, 13, 16]. Another domain dealing with dynamical fitness landscapes is co-evolution [6, 11] where the fitness of an individual depends on the phenotype of the individuals in the current population. This is qualitatively different to the robotic experiments, because here the fitness landscape depends on the position of the robot in the environment and the state of the environment. Dynamic fitness landscapes can be modeled as oscillating functions (e.g. oscillating Fletcher-Powell function [9] or oscillating NK-landscapes [17]) which depend on time, only.

Note that in general the change of a dynamic fitness landscape can not be regarded as a function over time which maps t to a fitness function, as in [9, 17]. In our case the change of the fitness landscape depends heavily on previous fitness evaluations and therefore on robot actions. The landscape changes because the robot moves around and thus encounters situations of different difficulty. The change of the fitness landscape should be regarded as a dynamical process influenced by the executed individuals controlling the robot.

In this paper we introduce a simple method how the change of the fitness landscape over time can be visualized and analyzed based on reference individuals. The method is demonstrated by applying it to the evo-

[1] An **individual** is defined as an element of the search space.

[2] A **fitness landscape** is the relation between individuals and fitness values. The fitness value characterizes the quality of an individual which should be optimized. This notion of fitness should not be confused with the biological notion of fitness.

lution of a controller for a random morphology robot which should move the robot forward as fast as possible. We found that the fitness landscape changes drastically over time. It is also shown that there may be regions in time where good individuals can not be discriminated from better ones. Potential implications for the design of evolutionary algorithms for dynamical fitness landscapes are discussed.

2 RANDOM MORPHOLOGY ROBOT

A random morphology robot (RM robot) is composed of a couple of servo motors which are connected arbitrarily (Fig. 1 and 2) [3, 4]. Thus, its morphology is called random. The hardware structure can be changed very easily and can even be made arbitrarily complex by simply adding more servo motors and connections. Simulations of evolving morphologies in virtual environments [7, 15] have motivated the RM robot. It is also related to bio-inspired robots which consist of many similar and simple interconnected components [18].

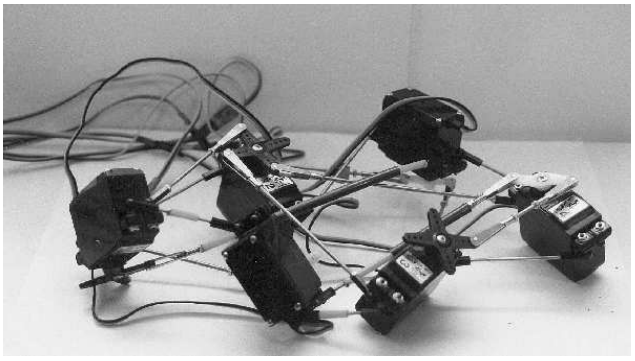

Figure 1: The random morphology robot (RM robot) without movement measuring cart.

Actuators: The RM robot is composed of a couple of conventional RC servos which are connected randomly. These devices possess a complete servo system including: motor, gear box, feedback device, servo control circuit, and drive circuit. The connections are made by brass poles available for hobby modeling. They can be easily connected to the servos, thus one can set up or change an architecture quickly, which is useful for evolutionary experiments in hardware.

Sensors: Movement of the RM robot is measured by a computer mouse device, mechanically connected to the robot (Fig. 2). This device allows precise measurement of motion in the 2-D plane. The positional error is about $1mm$.

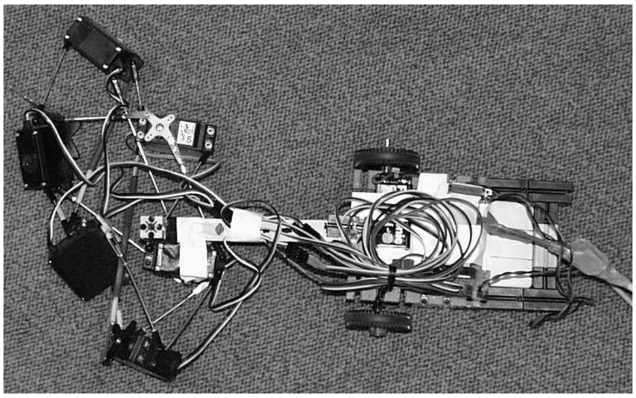

Figure 2: The random morphology robot from top with movement measuring cart. This picture has been taken during the experiments described here.

Control: The servos are controlled by a pulse signal generated by a micro-controller connected to the host computer by a serial RS232 interface. The host is a PC running LINUX which we found well suited for this task. It is fast enough even without a real-time kernel. A piece of interface software was written to control the servos via the serial RS232 line.

3 EVOLVING ROBOT CONTROLERS

To evolve control programs we applied GP [2, 8] as a learning method. A variant of stochastic sampling as in [14] has been used where the fitness evaluation for an individual is performed by allowing the individual to control the robot only for a short time (here, about 10 seconds).

3.1 THE GENETIC PROGRAMMING SYSTEM

Here, a simple tree-based steady-state GP algorithm is applied. An individual controls the robot only for short time by executing the individual a few times (see Sec. 3.3). Its performance is measured while the robot advances in the desired direction. The fitness is evaluated during the tournament selection. In a **tournament** four individuals are selected randomly. Their fitness is evaluated by calling the procedure $evalFit()$ (Sec. 3.3). The two worst performing individuals are replaced by offsprings from the two best performing individuals. As in conventional GP one subtree in each parent is selected randomly and exchanged for recombination. Nodes near the root have a higher probability to be selected as crossover points in order to reduce the number of neutral recombinations. After recombination each offspring is mutated with a probability of 0.9. For this one randomly selected node from the offspring is replaced by a node randomly selected from the set of nodes with the same arity. So, a terminal is always replaced by another terminal.

3.2 INDIVIDUAL SYNTAX AND SEMANTICS

An individual is represented by a parse tree with the function set shown in Tab. 1. The arithmetic operators have arity two, DELAY and SETSERVOx have arity one. The leafs are float constants from the interval $[-150, 150]$. When an individual I is executed by the interpreter ($exec(I)$) its return value is discarded. Its output are side effects generated by DELAY and SETSERVOx. The operation (DELAY t) causes the interpreter to pause for t milliseconds. The operation (SETSERVOx a) sets the servo position of servo x to angle a. The operation does not wait until the servo motor has adjusted the servo to the new position but returns immediately so that different servos can be set virtually at once by successive SETSERVOx operations.

Similar to the artificial ant example ([8], p. 147) an individual is executed in the following loop which is used for fitness evaluation:

$execInd(I, t_{max}, n_{max}) :=$
(1) $n_{rep} \leftarrow 0$ and set real time clock to zero
(2) exec(I)
(3) $n_{rep} \leftarrow n_{rep} + 1$
(4) if $n_{rep} < n_{max}$ and real time $< t_{max}$ goto 2
(5) return (real time $/n_{rep}$, n)

The procedure $execInd(I, t_{max}, n_{max})$ executes the individual I until the maximum time t_{max} or the number of allowed executions n_{max} is exceeded. It returns the number of executions n_{rep} and the average execution time per individual in seconds.

3.3 FITNESS EVALUATION

To evaluate the current fitness $f = evalFit(I)$ of an individual I the robot is initialized by executing the individual a few times for about 3 seconds without measuring the movement (step (1)). Then the individual is executed for about 8 seconds while the distance moved is measured (steps (2)-(4)). The resulting fitness is a weighted sum of the covered distance and speed. The algorithm for evaluating the fitness of an individual I reads:

$evalFit(I) :=$
(1) $execInd(I, 3s, 4)$
(2) $(x_1, y_1) \leftarrow getMousePosition()$
(3) $(\Delta t, n_{rep}) \leftarrow execInd(I, 8s, 20)$
(4) $(x_2, y_2) \leftarrow getMousePosition()$
(5) $s_f \leftarrow (y_1 - y_2)/n_{rep}$, $s_s \leftarrow (x_1 - x_2)/n_{rep}$
(6) $v_f \leftarrow s_f/\Delta t$, $v_s \leftarrow s_s/\Delta t$
(7) return $f \leftarrow (\frac{1}{3}s_f + v_f) + \frac{1}{8}(\frac{1}{3}s_s + v_s)$

where s_f and s_s is the distance covered by one individual forward and sidewards, respectively. Δt is the real time (measured in seconds) that is required for one execution of the individual, v_f and v_s are the speed the robot moves forward and sidewards, respectively. The procedure $getMousePosition()$ returns the current mouse position.

3.4 REFERENCE FITNESS

From diagrams showing the population fitness over time (e.g. best or average fitness) it is not possible to derive with confidence whether the system achieves any improvements. This is because the fitness landscape constantly changes. In order to overcome this problem we have defined a **reference fitness** $f_{ref}(t)$, which is the fitness of a well performing reference individual that is evaluated after each tournament.

Based on reference fitness the **relative fitness** $f_{rel}(I, t)$ of an individual can be calculated as

$$f_{rel}(I, t) = \frac{f(I, t)}{f_{ref}(t)} \tag{1}$$

where $f(I, t)$ is the fitness of the individual measured nearly at the same time as the reference fitness is measured and $f_{ref}(t) > 0$.

It is important to choose good programs as reference individuals possessing high fitness values to reduce fluctuations of the relative fitness.

3.5 BEHAVIOR OF THE GP SYSTEM

In [3, 4] we have shown that tree-based and linear GP is able to evolve control programs which move the robot. Figure 3 shows fitness over time of an experiment with the setting given in Tab. 1. As it can be observed in Fig. 3 absolute fitness values and relative fitness are fluctuating heavily which is typical for all experiments.

4 DYNAMIC CHARACTERISTICS OF THE FITNESS LANDSCAPE

This section describes the result from an investigation of dynamic properties of the changing fitness land-

Objective	Find a program that moves the robot straight on as fast as possible
Raw fitness	The sum of pixels the mouse pointer travels in a desired direction minus the sum of pixels the mouse pointer travels in the opposite direction. (See text.)
Fitness	Weighted sum of speed and distance traveled. (See text.)
Terminal set	CONST (random constants)
Function set	ADD, SUB, MUL, DIV, DELAY, SETSERVO0, SETSERVO1, SETSERVO2, SETSERVO3, SETSERVO4, SETSERVO5, SETSERVO6
Population size	$M = 50$
Initialization method	half-and-half, max nodes: 200, max depth: 50
Maximal number of nodes	$l_{max} = 500$ nodes
Probability of mutation	$p_m = 0.9$
Number of nodes mutated	$n_{mut} = 1$ node
Probability of crossover	$p_c = 1.0$
Probability of reproduction	$p_r = 0$
Tournament size for genetic operators	$T_r = 4$
Termination criteria	decision by experimenter

Table 1: Koza tableau of the evolution of motion control programs for the experiments reported here.

scape. It is not intended to characterize the landscape according to the search space (e.g. ruggedness or causality) but to characterize its time-dependence.

4.1 METHOD

For our analysis the following method has been applied:

Step 1: Select **reference individuals** from different sources, e.g. from normal evolutionary runs or random populations. It is important that (at least some of) the individuals have high fitness values.

Step 2: Perform experiments each with two reference individuals I_1, I_2 selected from this set by evaluating their fitness alternating with the same setting as have been used in the evolutionary experiments:

$measureFit(I_1, I_2) :=$

(1) $t \leftarrow 0$
(2) $F(I_1, t) \leftarrow evalFit(I_1)$
(3) $t \leftarrow t + 1$
(4) $F(I_2, t) \leftarrow evalFit(I_2)$
(5) $t \leftarrow t + 1$
(6) goto (2)

Step 3: Use obtained data $F(\ldots)$ to characterize the dynamic property of the fitness landscape as it will be demonstrated in the following sections.

4.2 REFERENCE INDIVIDUALS

For the experiments reported here we took seven reference individuals, arbitrarily: three reference individuals from two evolutionary experiments (speed2-long and speed2-1) with the same setting as shown in Tab. 1 and four from a sequence of randomly generated individuals. We assured that the strategy of the individuals to move the robot are different. And we intentionally selected well and poor performing individuals to get a diverse spectrum. The fitness values given in the following table are the fitness values obtained in the experiments. A comparison of the quality of two individuals should not be based on these values.

name	f	source	avg. performance
ind1	5.14015	speed2-long	very good
ind2	4.02222	speed2-1	good
ind3	2.91667	rand	good
ind4	2.08333	rand	poor
ind5	1.23958	speed2-long	poor
ind6	0.266667	rand	very poor
ind7	0791667	rand	very poor

4.3 RESULTS

The results described in this section are based on 17 runs of the algorithm $measureFit(\ldots)$ (Sec. 4.1) with 9 different pairings of the reference individuals ind1

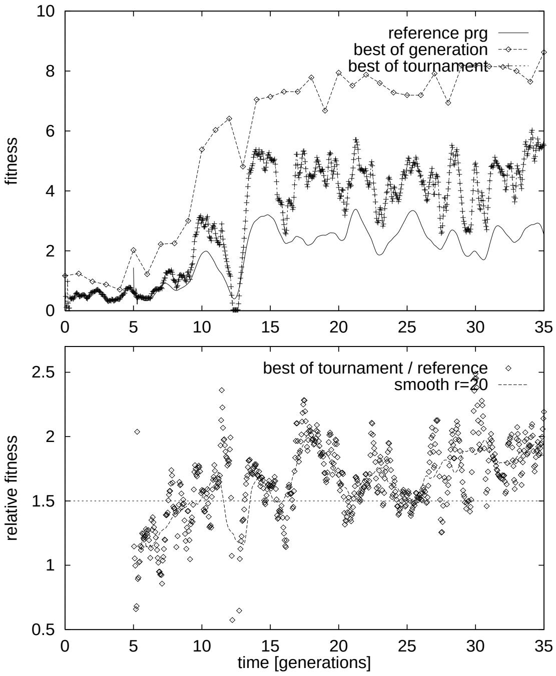

Figure 3: Smoothed fitness development over time (top) and relative fitness of a long run [4]. The reference individual has been taken as the best individual of generation 5 (Series 3, run speed2-long).

- ind7. For these experiments a total of 7467 fitness evaluations have been performed. A typical run of 500 fitness evaluation lasts about 2 hours. To present and discuss the data obtained typical graphs are shown below. The data of all runs is available via WWW from the address given below.

4.3.1 Very Good vs. Good

Figure 4 shows a typical run with ind1 (very good) and ind2 (good). The robot performs a trajectory as illustrated in Fig. 6 which is also typical for 90% of all runs. During the robot movement the fitness landscape changes dramatically. Overall it becomes more difficult because the cable stretches and the robot turns around. This latter fact has a strong influence because carpet is not "isotropic". In the orientation at position (P1) in Fig. 6 friction between servos and the carpet is much higher than in the orientation at position (P3) so that it is easier to move.

When the robot has reached position (P3) fitness evaluation process is halted (after step (4) of algorithm $evalFit(\ldots)$) and the robot is relocated to position (P1) by hand. This is indicated in the diagrams by a sudden increase of the fitness values.

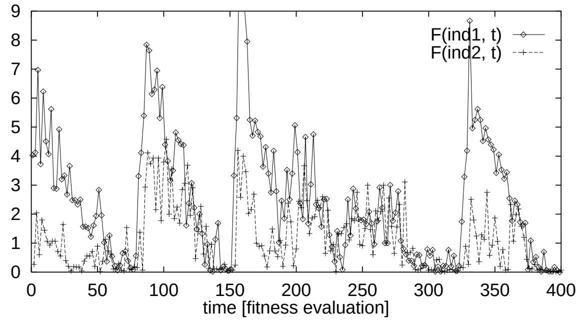

Figure 4: Fitness development over time of individuals ind1 and ind2. The robot has been relocated to position (P1) three times in this figure. Series B, run2-4.

4.3.2 Good vs. Poor

Figure 5 shows a typical run of a well performing and poorly performing reference individual. The good individual (ind3) can be clearly discriminated from the bad individual (ind4) in nearly all situations.

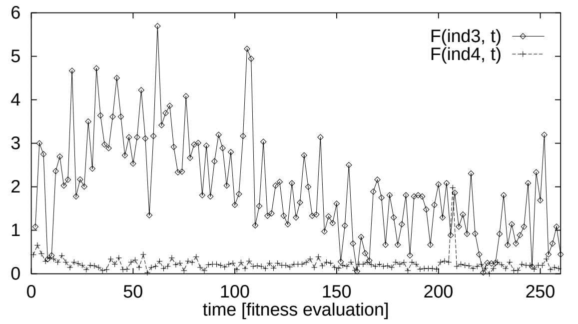

Figure 5: Fitness over time of individuals ind3 and ind4. The graph represents one robot trajectory without any relocation. Series B, run3-2.

4.3.3 Good vs. Good

Figure 7 shows the differential fitness over time of two well-performing individuals (ind2 vs. ind3) with similar average performance. The points H1 to H7 in Fig. 7 mark the relocation of the robot to its starting position (P1). Figure 6 shows a typical movement of the robot between $t = 100$ and $t = 240$, according to Fig. 8. The movement is subdivided into four phases represented by position (P1) to (P4) in Fig. 6:

phase	meaning
P1	start
P2	a curve with an angle of 180 degree
P3	a route straight on
P4	low fitness values caused by the stretched cable and orientation

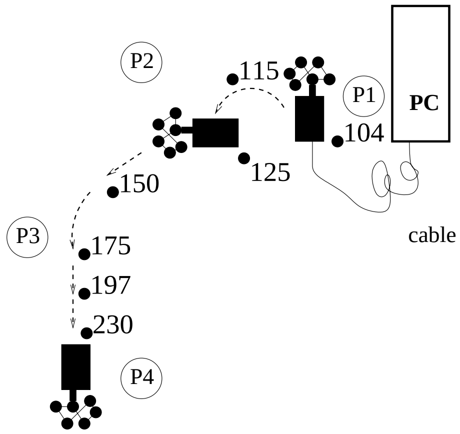

Figure 6: Typical behavior of the RM robot during the experiments. The numbers are time steps refering to Fig. 8

For analysis we now concentrate on a typical trajectory from (P1) to (P4) sketched in Fig. 6. Figure 8 shows an enlarged window of the fitness development between $t = 100$ and $t = 240$. In the first phase ind3 is superior to ind2. If the situation becomes more difficult two qualitatively different behaviors can be observed. For example, between $t = 170$ and $t = 190$ the fitness of both is very low and they cannot be discriminated. In the interval $t = 190$ to $t = 210$ ind2 is superior to ind3 which has a fitness near zero.

Looking at the movements during this time the high fitness of ind3 corresponds with the curve phase at (P2). Better fitness values of ind3 corresponds with the straight-on phase at (P3). This is a typical example of two individuals performing differently in different situations, e.g. ind3 when the situation favors curves, ind2 when a straight-on movement is favored, which depends on the cable and carpet.

Expanded to the complete run (showed in Fig. 7) this assumption has been confirmed in several sections. Superior fitness values of ind3 correlate with curve phases and superior values of ind2 with straight-on phases.

This shows that even in this simple experimental setting there are regions in search space where the fitness landscape can be described as oscillating, which motivates to model fitness landscapes as (partly) oscillating functions [9, 17].

5 CONCLUSION

The fitness landscape changes because the robot moves around and thus encounters situations of different difficulty. We have demonstrated that in our case the

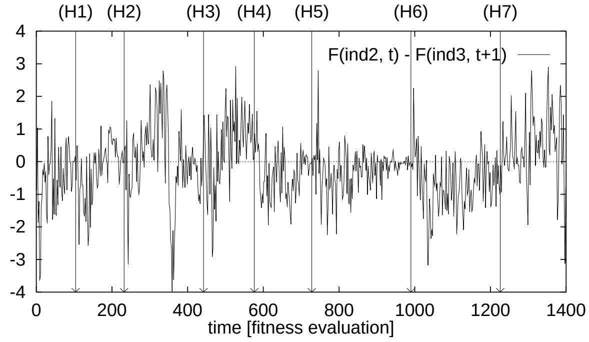

Figure 7: Differential fitness development of two well performing individuals (ind2 vs. ind3). Series B, run6-1.

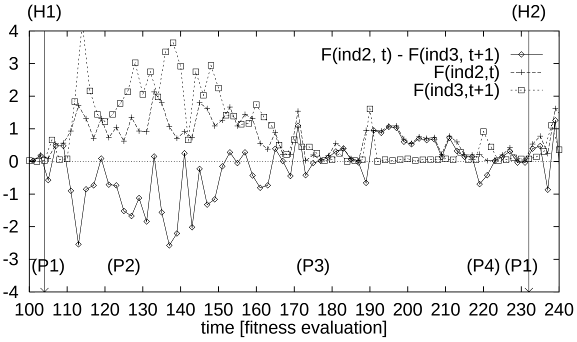

Figure 8: Fitness and differential fitness development of ind2 vs. ind3. Enlarged window of Fig. 7. Series B, run6-1.

fitness landscape changes drastically so that it is difficult to compare individuals based on fitness values evaluated at different points in time. In a difficult situation, a good individual cannot be discriminated from an individual with an overall lower performance. In easy situations where high fitness values can be obtained good individuals can be compared based on the relative fitness, because the noise level is lower and discrimination is easier. There are also regions where the fitness landscape can be described as oscillating which makes comparisons based on the relative fitness problematic.

The implication for the design of on-line evolutionary algorithms is: There are regions in time where the fitness landscape does not allow to discriminate well from better performing individuals. These situations can be detected by measurements based on reference individuals. In these regions, evolution should be “turned off” and can be turned on again when the robot has left these regions. It may also be helpful to use not only one reference individual (as shown in Fig. 3) but two or more which allows to measure how well individuals can be discriminated in the current situation and whether the landscape oscillates.

Acknowledgments

Support has been provided by the DFG (Deutsche Forschungsgemeinschaft) under grant Ba 1042/2-2 and project B2 of SFB 531. We would also like to thank Andreas Bürgel for building parts of the robotic hardware and Robert Keller and the reviewers for their helpful comments.

Supplement Material

To ensure reproducibility more detailed information, source code, raw experimental data and hints for setting up the hardware are available from:

http://ls11-www.cs.uni-dortmund.de/alife/rmrobot

References

[1] Thomas Bäck. *Evolutionary Algorithms in Theory and Practice.* Oxford University Press, New York, 1996.

[2] Wolfgang Banzhaf, Peter Nordin, Robert E. Keller, and Frank D. Francone. *Genetic Programming - an Introduction.* dpunkt, Heidelberg and Morgan Kaufmann, San Francisco, CA, 1998.

[3] Peter Dittrich, Andreas Bürgel, and Wolfgang Banzhaf. Learning to move a robot with random morphology. In Phil Husbands and Jean-Arcady Meyer, editors, *First European Workshop on Evolutionary Robotics*, pages 165–178. Springer, Berlin, 1998.

[4] Peter Dittrich, Andreas Bürgel, and Wolfgang Banzhaf. Random morphology robot - a test platform for online evolution. *submited to Robotics and Autonomous Systems*, 1998.

[5] Ingman Harvey, Phil Husbands, Dave Cliff, Adrian Thompson, and N. Jakobi. Evolutionary robotics: the sussex approach. *Robotics and Autonomous Systems*, 20:205–224, 1997.

[6] Stuart A. Kauffman. *The Origins of Order: Self-Organization and Selection in Evolution.* Oxford University Press, Oxford, 1993.

[7] Maciej Komosiński and Szymon Ulatowski. Framsticks - artificial life. In C. Nědellec and C. Rouveirol, editors, *ECML 98 Demonstration and Poster Papers*, Chemitzer Informatik-Berichte, Chemnitz, pages 7–9, 1998.

[8] John R. Koza. *Genetic Programming.* MIT Press, Cambridge MA, 1992.

[9] Frank Kursawe. Following a moving optimum with an ES. Internetserver of the Chair Systems Analysis, University of Dortmund, http://ls11-www.informatik.uni-dortmund.de/people/kursawe/demos.html, 1997.

[10] Wei-Po Lee, John Hallam, and Henrik Hautop Lund. Learning complex robot behaviours by evolutionary approaches. In *6th European Workshop on Learning Robots*, pages 42–51, Brighton, UK, 1-2 August 1997.

[11] Kristian Lindgren and Mats G. Nordahl. Evolutionary dynamics of spatial games. *Physica D*, 75:292–309, 1994.

[12] Maja J. Matarić and Dave Cliff. Challenges in evolving controllers for physical robots. *Robotics and Autonomous Systems*, 19(1):67–83, 1996.

[13] Stefano Nolfi, Dario Floreano, Orazio Miglino, and Francesco Mondada. How to evolve autonomous robots: Different approaches in evolutionary robotics. In Rodney A. Brooks and Pattie Maes, editors, *Artificial Life IV*, pages 190–197, Cambridge, MA, 1994. MIT Press.

[14] Peter Nordin and Wolfgang Banzhaf. An on-line method to evolve behavior and to control a miniature robot in real time with genetic programming. *Adaptive Behaviour*, 5(2):107–140, 1996.

[15] Karl Sims. Evolving 3D morphology and behavior by competition. *Artificial Life*, 1(4):353–372, 1994.

[16] Luc Steels. Emergent functionality in robotic agents through on-line evolution. In Rodney A. Brooks and Pattie Maes, editors, *ArtificialLife IV*, pages 8–16, Cambridge, MA, 1994. MIT Press.

[17] Claus O. Wilke. Evolution in time-dependent fitness landscapes. Technical Report IR-INI 98-09, Institut für Neuroinformatik, University of Dortmund, Ruhr-Universität Dortmund, Institut für Neuroinformatik, 44780 Bochum, 1998.

[18] Hiroshi Yokoi, Wenwei Yu, Jun Hakura, and Yukinori Kakazu. A morpho-functional machine: An artificial amoeba based on the vibrating potential method. In C. Adami, R. Belew, H. Kitano, and C. Taylor, editors, *Artificial Life VI*, pages 477–482, Cambridge, MA, 1998. MIT Press.