Applied Intelligence, **17**(1):89-100, 2002

Evolving Receptive-Field Controllers for Mobile Robots

Ralf Salomon
Department of Electrical Engineering and Information Technology
Institute of Applied Microelectronics and Computer Science
University of Rostock, 18051 Rostock, Germany
ralf.salomon@etechnik.uni-rostock.de
Phone: +49-381-498 35 29; FAX: +49-381-498 36 01

Abstract. The use of evolutionary methods to generate controllers for real-world autonomous agents has attracted recent attention. Most of the pertinent research has employed genetic algorithms or variations thereof. Recent research has applied an alternative evolutionary method, evolution strategies, to the generation of simple Braitenberg vehicles. This application accelerates the development of such controllers by more than an order of magnitude (a few hours compared to more than two days). Motivated by this useful speedup, this paper investigates the evolution of more complex architectures, receptive-field controllers, that can employ nonlinear interactions and, therefore, can yield more complex behavior. It is interesting to note that the evolution strategy yields the same efficacy in terms of function evaluations, even though the second class of controllers requires up to 10 times more parameters than the simple Braitenberg architecture. In addition to the speedup, there is an important theoretical reason for preferring an evolution strategy over a genetic algorithm for this problem, namely the presence of epistasis.

Keywords: Autonomous Robots, Evolutionary Algorithms, Radial Basis Functions, Scalability

1. Introduction

Mobile robots, as considered in this paper, are embodied systems that autonomously behave in the real world [5, 22]. An agent is consid-

ered *autonomous*, if it behaves on its own without any human control. Research on autonomous agents aims at understanding intelligent behavior as a system-environment interaction. Similar to living things that have to interact with their environment in order to survive, mobile robots are intended to do some useful work. Consequently, autonomous agents are urged to act; sitting on a spot would not be sufficient. In this context, moving around while avoiding obstacles is a fundamental behavior that is investigated in this paper. A good overview of some case studies and special design issues can be found in [17, 22].

Mobile robots are equipped with sensors and effectors, such as infrared sensors, ultra sonars, CCD cameras, motors, and grippers. An agent coordinates its actions by means of a control structure. As opposed to other approaches, autonomous agents do not explicitly employ a world model whatsoever. Instead, autonomous agents normally employ control structures that provide a tight coupling between sensors and effectors. Such control structures can be easily implemented as a neural network. Section 2 presents a brief discussion of a simple control architecture, the Braitenberg vehicle [4], and provides a summary of approaches that optimize such control structures by means of evolutionary algorithms.

Evolutionary algorithms are a framework that comprises many different algorithms, such as genetic algorithms (GAs), evolution strategies (ESs), evolutionary programming (EP), and other variants. Each of these algorithms is designed along different methodologies. Despite their differences, all evolutionary algorithms are heuristic population-based search procedures that incorporate random variation and selection. Typically, such population-based search procedures generate offspring in each generation. A fitness value (defined by a fitness function) is assigned to each offspring. Depending on their fitness, each population member is given a specific survival probability. Most evo-

lutionary algorithms can be described in the following generic form:

Step 0: Initialize the population's individuals and evaluate each individual's fitness

Step 1: Select the parents according to a selection scheme (e.g., roulette wheel, linear ranking, truncation selection)

Step 2: Recombine and mutate selected parents with a specific operators

Step 3: Go to Step 1

Examples presented in [8, 9, 20] show that GAs often require a large number of fitness evaluations with respect to the number of parameters, and the results of other research [24, 25] strongly indicate that the presence of epistasis, which refers to a nonlinear interaction of parameters with respect to the fitness fucntion and is omnipresent in real-world controllers, can significantly degrade the performance of GAs. Section 2 summarizes these problems, and presents a comparison with a ES-based approach [26] which yields a speedup of more than an order of magnitude as compared to published GA-based approaches.

Encouraged by this speedup, this paper focuses on the evolution of more complicated control architectures by means of ESs. These control architectures allow for more complex, nonlinear behavior than the simple Braitenberg architecture, as described in Section 2, in which the sensor activities have only a linear effect on the motor activations.

The experimental setup, including the methods used, is described in Section 4. All experiments have been performed on the *Khepera*^{TM} robot, whose technical description can be found in Appendix A. Section 3 is devoted to a detailed description of the receptive-field architectures, and experimental results are presented in Section 5. The ES yields the same efficacy on receptive fields, even though these controllers require up to 10 times more parameters than simple Braitenberg controllers. This efficacy is of great practical relevance, since the experiments are

to be done in physical robots that dynamically interact with the real world, and since it encourages the application to more complex control structures, which are currently intractable.

In Section 6, some results are further discussed, and Section 7 provides a conclusion.

2. Background

For a given mobile robot, the designer has to decide on how to implement an appropriate control structure. The implementation can be done, for example, in the C-programming language (if-then-else rules) or a set of Prolog facts and rules. Unfortunately, these implementations can quickly grow in size if the set of possible input states becomes large. In contrast, Braitenberg [4] has suggested rather simple control architectures, in which the sensors are directly coupled with the motors via some inhibitory and excitatory connections.

Figure 1 shows an example, which is inspired by a Braitenberg type-3C vehicle. Such a vehicle achieves object avoidance in the following way. When sensing an obstacle, sensors with a high proximity activation accelerate the motor on the sensors' side whereas these sensors slow down the motor on the opposite side. By this mechanism, the presence of an obstacle leads to different motor speeds, which results in particular trajectories. Depending on the activation of all proximity sensors, the robot either turns, spins on the spot, or even backs up.

More formally, the activation of the left motor M^l is calculated by the following formula

$$M^l = \sum_{i=1}^{8} IR_i w_i^l + w_0^l \ , \tag{1}$$

where IR_i denotes the activation of the proximity sensor i and w_i^l denotes the weight that connects proximity sensor IR_i with the left

motor. The weight w_0^l represents the idle activation of the left motor. This "bias" weight is responsible for the robot's forward motion in the absence of any obstacle. The calculation of the right motor's activation is similarly given by

$$M^r = \sum_{i=1}^{8} IR_i w_i^r + w_0^r \ . \tag{2}$$

Given a particular architecture, e.g., a Braitenberg type-3C controller, the remaining task is to determine the weights w_i^l, w_i^r, w_0^l, and w_0^r such that the robot is moving around while it avoids obstacles.

Among other approaches, such as trial and error or backpropagation learning, evolutionary algorithms can be used to determine these parameters. Evolutionary algorithms have the advantage that the fitness evaluation can be easily done in situ, i.e., on the real robot, whereas other approaches, such as neural network learning, often require additional efforts, such as collecting a set of training patterns.

The pertinent literature reports several applications that use GAs for the evolution of various control architectures. Examples can be found in [6, 8, 9, 14, 15, 20]. Even though these GA-based approaches successfully evolve controllers, certain problems can be identified with GAs. First, GA performance is perhaps a bit low with respect to the number of parameters. For the evolution of Braitenberg controllers with fewer than 20 parameters, GAs have required more than 60 hours when function evaluations are performed on real robots [8, 9, 20]. To speed up the evolution process, the number of infrared sensors can be reduced [20], or the number of bits per connection can be chosen as small as possible. In [15], for example, the search space is significantly reduced by encoding real-valued parameters by merely three or four bits. However, these reduction strategies are not very satisfying, since the long-term goal of autonomous agent research is to use more complex input devices that consist of many more sensors, which yield much more accurate sensor values. Second, it is well known that the presence of

epistasis can significantly degrade the performance of GAs. Epistasis describes the interaction of parameters with respect to the fitness of an individual. A brief discussion of the effect of epistasis can be found in Appendix B.

It has been pointed out [27] that small mutation rates in the range of $p_m = 1/n$, with n denoting the number of parameters, are the major source of the epistatic sensitivity; small mutation rates and uniform recombination bias the search along coordinate axes. Increasing the mutation rates does not solve this problem [27]. The problem in the field of autonomous agents is that epistasis is omnipresent in most real-world controllers and that GAs appear to be very sensitively with respect to epistasis. For example, mutating only the bias weight w_0 of a Braitenberg controller normally leads to a decreased fitness; the robot tends either to crash into obstacles or circle on the spot. In most control architectures, all weights should be mutated simultaneously. This interpretation is in concordance with the analysis presented in [25, 27].

The application of mutations to all parameters can be found in evolution strategies (ESs) [23, 28] and evolutionary programming (EP) [11, 10]. ESs and EP are very similar, and when applied to the evolution of controllers for autonomous agents, it can be expected that both schemes yield very similar results. The following description focuses on ESs.

A $(\mu \dot{+} \lambda)$-ES maintains a population of μ parents and generates λ offspring in each generation. A (μ, λ)-ES indicates that μ parents generate λ offspring, and μ new parents are selected only from the λ offspring, whereas in a $(\mu + \lambda)$-ES, the μ parents are selected from the union of parents and offspring. For further details see [2]. Currently [2], the (μ, λ)-ES is recommended, especially in noisy environments.

In contrast to traditional GAs, ESs encode each real-valued parameter as a floating-point number, and they apply mutation to *all*

parameters simultaneously, i.e., $p_m = 1$. Mutations are typically implemented by adding $(0, \sigma)$-normal distributed random numbers. A key feature of ESs is that they self-adapt the mutation strength σ. In its simplest form, each individual is enhanced by a step size σ, which is global with respect to the individual's parameters and local with respect to the population members. Each offspring inherits its step size from its parent(s). This step size is typically modified by taking the product of itself with a lognormal random number prior to mutation. By this means, the step size is self-adapted: those offspring survive that have the best adapted step size. For further details of different step size schemes see [2, 28]. In more elaborate forms, ESs maintains one step-size for each parameter or even maintains a whole $n \cdot n$ matrix with $n(n-1)/2$ independent parameters to generate correlated mutations. An overview of these more elaborate methods can be found in [2, 28]. In addition, ESs can use the same recombination schemes as GAs [2].

A comparison of ESs and GAs when applied to the evolution of Braitenberg controllers has already been presented in [26]. There, a simple (3,6)-ES was used to evolve constrained as well as unconstrained controllers as defined by equations (1) and (2). In a *constrained* controller, both sides have identical connections according to $w_0^l = w_0^r$ and $w_i^l = w_{(14-i) \bmod 8+1}^r$. This constraint is motivated by the symmetry of the robot's morphology including sensors, motors, and the controller.

Figure 2 (taken from [26]) shows both the fittest agent and the population average when evolving a constrained controller. It can be seen that fewer than 40 generations are sufficient for the evolution of reasonable controllers. After 30 generations, with the fittest controllers, the robots perform up to three complete laps in the arena. It can also be seen that, at the beginning, some controllers have a negative fitness, which results from crashing backwards into the wall.

The reduced search space of constrained controllers speeds up the evolutionary process by a factor of two and also leads to solutions with a higher fitness [26].

In comparison, the GAs used in [8, 9, 20] had a typical population size of 80 individuals and required about 50 to 100 generations. By contrast, the (3,6)-ES used in [26] required only 40 generations while generating only six offspring per generation. A comparison shows that the ES yields a speedup of more than an order of magnitude over the GA in terms of the number of function evaluations. This speedup is of great practical relevance, since for all function evaluations, the GAs require approximately 66 hours, whereas the ES requires fewer than 3 hours.

3. Receptive-Field Controllers

Braitenberg architectures are very appealing, because they are very simple and defined by a relatively small number of parameters, which implies a low-dimensional search space. Due to their simplicity, however, a pure Braitenberg controller cannot compensate potential nonlinearities of the sensors and it cannot develop more complicated behavioral patterns. This section describes another artificial neural network model, receptive fields, that offer the option of going beyond these limits. It should be noted, however, that receptive fields require many more parameters to be adjusted by the evolutionary process.

Artificial receptive fields are inspired by neurons that exhibit very specialized responses to visual stimuli as can be found in the visual cortex for instance. Some of these neurons selectively respond with respect to the spatial position, whereas others selectively respond with respect to the orientation of a straight line. The idea of selective responses can be easily implemented by radial basis functions (RBFs). A general

overview of RBFs can be found in [18, 29]. Since receptive fields are not widely used in the field of autonomous agents and since section 5 presents important and encouraging results, the receptive-field control architecture is explained in greater detail here.

The dimension of a receptive field is given by the number of its input neurons. An example of a one-dimensional receptive field is presented in Figure 3. Each hidden unit i of such a receptive field maintains its center $\vec{c}_i$ and its width β_i. The activation o_i^h of each hidden unit is determined by

$$o_i^h = e^{\frac{-D^2}{\beta_i^2}} \ , \tag{3}$$

where D denotes the Euclidian distance between the unit's center $\vec{c}_i$ and the input vector consisting of those infrared sensors to which the receptive field is connected, i.e., $D = \parallel \vec{IR} - \vec{c}_i \parallel$; in the one-dimensional case, the distance is simply $D = \parallel c_i - IR_j \parallel$, where IR_j denotes the activation of the connected infrared sensor. The output unit of each receptive field i averages over all hidden units in the following form

$$o_i^o = \frac{\sum_j w_{ij}^o o_j^h}{\sum_j o_j^h} \ , \tag{4}$$

where w_{ij}^o denotes the weight that connects hidden unit j with the output unit o_i^o of the receptive field i. In this architecture, the weights w_{ij}^o are subject to the evolutionary process.

In a complete controller that consists of one-dimensional receptive fields, each motor side is connected to each infrared sensor via one receptive field. When using only the six frontal sensors, a controller for Khepera has to implement 12 receptive fields $o_{il}^o, o_{ir}^o, 1 \leq i \leq 6$ by using equation (4). In such a configuration, the hidden layers can be shared, which reduces the hidden layer, i.e., the o_i^h, to six fields. The activation of each motor side is then calculated as the sum of all six receptive

fields

$$M^l = \sum_{i=1}^{6} o_{il}^o$$
$$M^r = \sum_{i=1}^{6} o_{ir}^o \ . \qquad (5)$$

It is interesting to note that a receptive-field controller does not have any explicit "idle" weights w_0^l, w_0^r; the "idle" weights are distributed into all receptive fields.

Figure 4 presents a two-dimensional receptive field, which consists of 16 hidden units, four in each dimension. For such a controller, equation (4) has to be extended appropriately. A controller that considers Khepera's six frontal sensors consists of five two-dimensional receptive fields.

A complete architecture with three-dimensional receptive fields is presented in Figure 5. Each box represents one receptive field. With 4 units per axis, each box contains 64 hidden units and 64 three-dimensional centers $\vec{c}$. In case of a constrained controller, the evolution process has to determine $4 \cdot 64 = 256$ output connections w_{ij}^o. A comparison with the simple Braitenberg architecture shows (see Figures 1, 4, and 5) that, essentially, each Braitenberg connection is substituted by an entire receptive field.

4. Methods

The experimental setup considered here consists of three different components: (1) the arena in which the robot has to operate, (2) the definition of the fitness function, and (3) the parametrization of the ES used in the experiments. Both the arena and the fitness function have been chosen to be as close as possible to those proposed in other research [8, 9, 20].

Figure 6 shows the arena in which the robot has to move. The arena is of size 60x45 cm and the walls are made from wood with a height of 3 cm. The width of the corridors is chosen such that always at least one proximity sensor has a less-than-maximum value.

As already discussed, a mobile robot has to move forward while it avoids obstacles. In order to evolve good Braitenberg vehicles, the fitness function has to incorporate the motor speeds and the distance to obstacles. However, using only speed and distance is not sufficient. In such a case, a robot that is spinning on the spot with a high speed far away from any obstacle would have a high fitness. But such a robot would not do anything practical. Therefore, the fitness function is enhanced by a third term that favors straight movements by penalizing turns (see also [8, 9, 20]).

The fitness is measured as follows. At each time step t, both motor speeds V_l and V_r, as well as all eight proximity sensors IR_i are measured. The speed of the robot's center $V_t = (V_l + V_r)/2$, the penalty term $\Delta v_t = |V_l - V_r|$, and the sensor with the highest activation $\hat{IR} = \max_i IR_i$ are then calculated. The fitness contribution f_t at time t is determined by

$$f_t = V_t(1 - \sqrt{\Delta v_t})(1 - \hat{IR}_t) \ . \tag{6}$$

Finally, the total fitness is the sum over $t_{\max}$ (e.g., 240) time steps

$$F = \sum_{t=1}^{t_{\max}} f_t = \sum_{t=1}^{t_{\max}} V_t(1 - \sqrt{\Delta v_t})(1 - \hat{IR}_t) \ . \tag{7}$$

Between two time steps, Khepera moves with constant speed. As is explained in Appendix A, the 100 millisecond duration between two consecutive time steps is due to A/D conversions and data transmission between Khepera and the host.

The fitness function (7) is adopted from other research and could have been defined differently. The evaluation of different fitness func-

tions would be very interesting but is far beyond the scope of this paper.

The third component of the experimental setup concerns the parametrization, e.g., population size and selection pressure, of the ESs used in the experiments. For this design issue, the following issues were considered. Due to the noisy sensor readings (see also Appendix A), the fitness evaluation is very noisy. In addition to the sensory noise, the inevitable variations of the individual's starting position is a further source of noise. Furthermore, the environmental conditions cannot be held constant over a long time; this phenomenon is intrinsic to real-world applications.

From a noisy fitness evaluation, spurious candidates may evolve whose fitness value may be too high. Therefore, a non-elitist (μ, λ) selection scheme, also known as truncation selection, was chosen. Also, some candidates might have too low a fitness value, which suggests the use of a rather low selection pressure. For these reasons, a (3,6)-ES was chosen. A (3,6)-ES indicates that, in each generation, six offspring are generated from three parents and that the parents for the next generation are selected only from the six offspring. In this paper, the ES generates two offspring without crossover, two with uniform recombination, and two with intermediate recombination. The initial standard deviations σ_i were set to 0.5.

In all experiments discussed in the next section, each dimension of a receptive field consists of four units and the widths β of all units were equally set to $\beta = 0.1$. All centers $\vec{c}_i$ of a receptive field were equally distributed with respect to the input space, i.e., $[0, 1]^N$ with N denoting the dimensions. Therefore, a one-dimensional receptive field controller with four units per dimension consists of $6 \cdot 4 = 24$ free weights/parameters, and a two-dimensional receptive field controller consists of $5 \cdot 16 = 80$ weights/parameters, respectively.

The chosen parametrization is justified to a certain degree but also somewhat arbitrary. Certainly, the performance of the ES could be

improved by some additional tuning. However, extensive statistics are prohibitive due to time cost of the physical evaluation. Rather, the first trial should yield reasonable performance.

5. Results

This section reports some results when applying a (3,6)-ES to the evolution of one, two, and three-dimensional receptive-field controllers.

Figures 7 and 8 show representative runs of the evaluation of one and two-dimensional receptive fields respectively. Both figures present the fittest controller as well as the population average. As can be seen from a comparison with Figure 2, the fitness values are comparable with those of simple Braitenberg controllers.

The comparison also shows that the (3,6)-ES evolves good receptive-field controllers, i.e., a fitness around or greater than 25, in a considerably shorter time than Braitenberg controllers. This phenomenon is quite interesting considering the fact that receptive fields require many more parameters than Braitenberg controllers. Yet, the reasons for this behavior are unclear. After about 10-15 generations, most test candidates perform more than two complete laps in the arena.

For statistical purposes, Figure 9 and 10 present the average over six different runs when evolving two-dimensional receptive-field controllers. The averaging over six runs seems a rather small statistic. However, all experiments have been done on real robots with human supervision of each experiment. Since each run requires approximately two to four hours, larger statistics were not affordable. and a statistic consisting of only three runs is often standard in this application domain [8, 9]; in simulations [20], sometimes the average of 10 runs is presented.

Compared to Braitenberg controllers, receptive-field controllers exhibit a much smoother behavior. Such a smoother behavior cannot be

seen in the performance figures, but it is the impression of the observer when looking at the agent-environment interaction. The smoothness of a particular behavior cannot be seen in any of the presented performance figures, since the smoothness is not expressed in the fitness function. Since the ES optimizes only the fitness function, candidates are favored that circle around with a high speed. But smoothness can be observed and expressed in the following way. A Braitenberg controller might travel very fast until it stops in front of an obstacle. It then backs up, turns, and proceeds with a high speed on its path. A receptive-field controller, by contrast, approaches the obstacle with a medium speed, does a small turn, and continuous on its path. Even though the latter controller exhibits (observer-based) a smoother behavior, both controllers can have the same fitness. For applications where smoothness is important the fitness function should be appropriately modified, but, as discussed above, the fitness function (7) has been adopted from previous research in order to allow for comparisons.

A different series of experiments focused on the use of sigmoidal activation functions in the output layer. Here, the motor activation is substituted by

$$M' = v_{\max}\left(\frac{2}{1+e^{-M}} - 1\right) \tag{8}$$

with $v_{\max}$ denoting the maximal motor speed, e.g., $v_{\max} = 15$. Equation (8) limits the motor speed to $\pm v_{\max}$. A typical run with 2-dimensional receptive fields and sigmoidal output units is presented in Figure 11. The performance is similar to receptive fields without sigmoidal output units, but the behavior, which cannot be seen from this figure, is much smoother. The robot slows down softly and turns without moving back and forth.

The receptive-field architecture allows the development of more complex behavior. Depending on the object-agent distance, for instance, a single receptive field can develop a positive as well as a negative contribution to the motor activation, which results in a behavior that cannot

be developed by a single Braitenberg connection. Also, a receptive field can easily compensate the nonlinearity of the infrared sensors, and a two-dimensional receptive field alone can develop a wall-following behavior.

6. Emergent Behavior

The autonomous agent research community places emphasis on the difference between preprogramed and *emerged* behavior. A behavior is considered emergent, if it results from a (dynamic) agent-environment interaction or from the interaction of independently-working control modules.

One well-known instance of a Braitenberg controller has the form $\vec{w}^l = (2.0,\ 0.6,\ 0.4,\ 0.3,\ -1.3,\ -1.1,\ -0.9,\ 0.0,\ 0.0)^T$, where the value 2.0 represents the idle activation w_0^l. Such a solution leads to turns in the presence of an object and the radius depends on the distance to the object and the object's position relative to the agent. With such a solution in mind, it can be expected that the evolution process yields very similar results.

It is interesting to note that in the experiments presented in [26], the evolution process often develops very different controllers with different behavioral mechanisms. Some of these solutions have the form $\vec{w}^l = (2.0,\ -0.2,\ 0.6,\ -0.1,\ -2.3,\ -1.5,\ -0.1,\ 0.0,\ 0.0)^T$. The connection w_1^l from the lateral sensor IR_1 to the left motor has a negative value -0.2 and the connection from sensor IR_2 has a positive value of 0.6. This particular weight vector implements a wall-following behavior. The presence of an object on the left-hand-side slows down the left motor. As a consequence, the robot turns towards the object. After turning, sensor IR_2 develops a positive influence on the left motor, which causes the robot to turn away from the object. After some steps,

the robot moves along an object/wall within a certain distance, where the effect of both sensors cancel each other, i.e., $IR_1 w_1^l = IR_2 w_2^l$.

Similar wall-following solutions have been emerged in the receptive-field controllers. Here, even more new solutions can be found. Some receptive fields provide positive as well as negative contributions to the motor activities, and others behave like the "normal" object avoidance network.

This wall-following behavior is by no means imposed by preprogramed mechanisms nor is it directly enforced by the fitness function. It is the result of the evolutionary process and the agent-environment interaction. Thus, the wall-following behavior is only one particular solution of the evolution/optimization process; by chance, the population has converged to this particular solution or basin of attraction, respectively. In other runs, the population converges to different solutions, sometimes to "normal" Braitenberg controllers and sometimes to other solutions. It should *not* be argued that this particular solution has certain other advantages. Selection is based solely on the controller's fitness with respect to the chosen fitness function; no other criteria are involved. The evolved controllers are results of the agent's embodiment, the environment, the sensors, the fitness function, etc. Thus, the final solution is partly a result of a random self-organization process.

7. Conclusions

This paper has discussed the practical application of evolution strategies to the evolution of receptive-field controllers for mobile robots. Mobile robots are autonomous agents with a simple control architecture, which is typically implemented as a neural network. Section 2 has summarized previous approaches for similar tasks. Most of these approaches have been GA-based. A comparative study presented in [26]

on simple Braitenberg controllers has shown that ESs yield a speedup of an order of magnitude. Encouraged by these results, this paper has focused on a more complex control architecture, receptive fields.

Receptive fields are inspired by neurons that can be found in, for example, the visual cortex. Braitenberg controllers are appealing, since they are simple and defined by a small number of parameters. Receptive fields, in contrast, require many more parameters and they are appealing because they can compensate for (electrical) non-linearities and they can engender more complicated behavior. With respect to this, it is very important that the applied (3,6)-ES yields the same or even slightly better performance when evolving receptive-field controllers. For this type of controller, previous attempts with GAs would be too inefficient, since a GA would require at least two orders of magnitude more real time.

The experiments have done on real robots, which require a considerable amount of time and effort. Resources (e.g., robots and personnel) are limited, and there was insufficient time for extended statistics in order to find an optimal tuning for the ES used; rather, the first trial should yield reasonable solutions in moderate time. For this purpose, both ESs and EP seem to be a very good candidates, since they self-adapt parameters, such as the step size or mutation probability. This allows to better embed the whole approach in more complex tasks.

Further research will be devoted to more complex control architectures for object avoidance, navigation, and manipulation as well as other neural controllers, which are designated to enhance the robots capabilities/competences

Appendix

A. Technical Details

Figure 12 shows the Khepera robot, which has been used in all the experiments presented in this paper. Its body is 32 mm high and 55 mm in diameter. The robot is equipped with eight infrared and eight ambient light sensors, as well as two motors. Khepera's sensors and motors are controlled by a Motorola 68331 micro controller. The robot can operate in two modes. In the first mode, a program is downloaded into the on-board memory, which allows Khepera to operate without any further hardware. In the second mode, Khepera is connected to a workstation via a serial link. In the experiments reported in this paper, the robot was controlled from a workstation and the ambient light sensors were not used. A detailed description of the robot and its electrical parts can be found in [16]. Due to A/D conversions and data transmissions to the host, reading the sensors and setting the motor speeds require approximately 100 milliseconds. During the 100 milliseconds time interval, the robot moves with constant speed.

The sensor readings of Khepera's eight infrared sensors are integer values in the range $[0, 1023]$. The interface normalizes the sensor readings such that the values are in the range $[0, 1]$. The sensors give reasonable input values for object distances between 10 mm and 60 mm. A sensor value of 1023 indicates that the robot is very close to an object, and a sensor value of 0 indicates that the robot does not receive any reflection of the infrared signal.

Khepera can control each motor independent of the other by sending commands to the robot. Valid speed values are integer values in the range $[-40, 40]$. The interface normalizes the motor speeds such that they are in the range $[-1, 1]$. Khepera's on-board PID controllers compensate the robot's dynamics and control the motors such that

the actual speed is as close as possible to the desired speed. It should be noted that rapidly changing motor commands induce additional dynamics, which can cause problems for the fitness evaluation. By means of attached wheel encoders, the robot measures the motor's real speed, which differ from the command setting in situations, where the robot cannot move. The real speeds can be obtained by sending special commands to the robot.

Khepera's limitation to integer speed values requires a floating-point-to-integer conversion. In order to avoid this artifact, the interface adds [0,1]-equal-distributed random numbers to the motor speeds at each time step. By this means, a controller output of, for example, 0.5 is converted into a "0-1-..." sequence. Without this special conversion, a continuous control output would be mapped onto a step function, which could cause further problems.

A major problem with Khepera is that the sensor readings are very noisy, which causes several difficulties for the fitness evaluation of a given controller. The effect of the noisy sensors to the fitness evaluation can be seen in Figure 13, which shows how the fitness contributions f_t are dynamically changing under constant environmental conditions, i.e., constant motor speeds, constant ambient light, and a constant position in the arena. Furthermore, the sensor readings are subject to several environmental conditions, such as the material the sensed object is made of, the object's surface, and the ambient light. During a long series of experiments, the environment cannot be kept constant.

B. Genetic Algorithm Performance

In the field of function optimization, many papers, e.g., [1, 2, 19], report efficient GA performance when applied the optimization of multimodal functions that contain millions of local optima but only one global

optimum. The performance that was achieved in these applications suggest that GAs easily escape from local optima and that they have very good global convergence. Some theoretical research [19] argues that particular GA variants yield a $O(n \ln n)$ computational complexity when applied to multimodal functions with a global structure. In [27] it has been shown that the $O(n \ln n)$ complexity is not due to some elaborated GA operators but due to the *separability* property of the fitness function studied. Most widely-used test functions, such as Rastrigin's function $f_6(\vec{x}) = \sum_{i=1}^{20} \left[x_i^2 - 10 \cos(2\pi x_i) + 10\right]$, can be decomposed into a sum of one-dimensional functions $f(\vec{x}) = \sum_i f^i(x_i)$. For separable functions, the normally recommended small mutation probability $p_m \approx 1/n$ [2, 7, 12, 19, 21] yields the $O(n \ln n)$ complexity.

It is well known that the presence of epistasis degrades the performance of GAs. In [27] the effect of epistasis was investigated by applying a rotation to the coordinate system. Rotation means that a linear and orthogonal transformation matrix $\boldsymbol{M}$ is applied such that $f_i(\boldsymbol{M}\vec{x})$ is calculated. A typical performance loss when applying the breeder genetic algorithm (BGA) [19] to Rastrigin's function f_6 is presented in Figure 14. While the BGA easily finds the optimum of the separable variant, it prematurely converges far away from the global optimum when a rotation is applied. The BGA is a GA variant that is tailored for the field of function optimization. It encodes each real-valued parameter as a floating-point number, it applies a mutation with a probability p_m, and it uses uniform recombination with a probability $p_r = 0.5$.

Results as well as theoretical analysis presented in [24, 25] strongly indicate that the independence of parameters is an essential prerequisite for successful performance of the GA, and that the computation complexity increases up to $O(n^n)$ for general multimodal functions. In other words, a GA with a small mutation rate becomes even less efficient than random search when optimizing multimodal functions in which

all parameters depend on each other. In [24], it is also discussed that increasing the mutation probability does not alleviate this problem.

In contrast to GAs, the performance of ESs is not effected by a rotation of the coordinate system. The reason is that EP and ESs simultaneously apply mutations to all parameters of each offspring, i.e., $p_m = 1$, and the mutation strength is self-adapted by an additional scheme. For further details and different self-adaptation schemes, see [2, 10, 23, 28].

Acknowledgements

This work was supported in part by a Human Capital and Mobility fellowship of the European Union, grant number ERBCHBICT941266. Thanks to Rolf Pfeifer, Christopher Lueg, and Peter Eggenberger for helpful discussion. Special thanks are due to David Fogel for detailed comments and editorial assistance.

References

1. T. Bäck, "Optimal Mutation Rates in Genetic Search," in *Proc. of the Fifth International Conference on Genetic Algorithms*, Morgan Kaufmann, San Mateo, CA, 1993.
2. T. Bäck and H.-P. Schwefel, "An Overview of Evolutionary Algorithms for Parameter Optimization," in *Evolutionary Computation*, vol. 1, no. 1, pp. 1-23, 1993.
3. H.-G. Beyer, "Toward a Theory of Evolution Strategies: On the Benefits of Sex – the $(\mu/\mu, \lambda)$ Theory," in *Evolutionary Computation*, vol. 3, no. 1, pp. 81-111, 1995.
4. V. Braitenberg, *VEHICLES, Experiments in synthetic psychology*. MIT-Press: Cambridge, Massachusetts, 1984.

5. R. Brooks, "A robust layered control system for a mobile robot," in *IEEE Journal of Robotics and Automation*, vol. 2, pp. 14-23, 1996.
6. D. Cliff, P. Husbands, and I. Harvey, "Evolving Visually Guided Robots," in *From animals to animats 2, Proc. of the Second International Conference on Simulation of Adaptive Behavior*, MIT Press, Bradford Books, Cambridge, MA, 1992.
7. K.A. De Jong, "An Analysis of the Behavior of a Class of Genetic Adaptive Systems," Ph.D. dissertation, University of Michigan, 1975.
8. D. Floreano, and F. Mondada, "Automatic Creation of an Autonomous Agent: Genetic Evolution of a Neural-Network Driven Robot," in *From Animals to Animats III: Proc. of the Third International Conference on Simulation of Adaptive Behavior*, MIT Press, Bradford Books, Cambridge, MA, 1994.
9. D. Floreano, and F. Mondada, "Evolution of Homing Navigation in a Real Mobile Robot," in *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, vol. 26, no. 3, pp. 396-407, 1996.
10. D.B. Fogel, *Evolutionary Computation: Toward a New Philosophy of Machine Learning Intelligence*, IEEE Press: Piscataway, NJ, 1995.
11. L.J. Fogel, "Autonomous Automata," *Industrial Research*, vol. 4, pp. 14-19, 1962.
12. D.E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Publishing Company: Reading, MA, 1989.
13. J.H. Holland, *Adaptation in Natural and Artificial Systems*, Ann Arbor: Michigan, University of Michigan Press, 1975.
14. I. Harvey, P. Husbands, D. Cliff, A. Thompson, and N. Jakobi, "Evolutionary Robotics: the Sussex Approach," in *Robotics and Autonomous Systems, Special Issues on "Practice and Future of Autonomous Agents"*, vol. 20, nos. 2-4, pp. 205-224, 1996.
15. S. Huber, H. Mallot, and H. Bülthoff, "Evolution of the Sensorimotor Control in an Autonomous Agent," in *From Animals to Animats 4: Proc. of the Fourth International Conference on Simulation of Adaptive Behavior*, MIT Press, Bradford Books, Cambridge, MA, 1996.
16. *Khepera Users Manual*, Laboratoire de microinformatique, Swiss Federal Institute of Technology (EPFL), 1015 Lausanne, Switzerland.
17. P. Maes, ed, *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back*, The MIT Press: Cambridge, MA, 1991.

18. J. Moody, and C. Darken, "Learning with Localized Receptive Fields," in *Proc. of the 1988 Connectionist Models Summer School*, Morgan Kaufmann Publishers, San Mateo, CA, 1989.

19. H. Mühlenbein, and D. Schlierkamp-Voosen, "The Science of Breeding and its Application to the Breeder Genetic Algorithm," in *Evolutionary Computation*, vol. 1, no. 4, pp. 335-360, 1993.

20. S. Nolfi, and D. Parisi, "Learning to Adapt to Changing Environments in Evolving Neural Networks," Technical Report 95-15, Institute of Psychology. National Research Council, Rome, Italy.
http://kant.irmkant.rm.cnr.it/public.html. 1995.

21. M.A. Potter, and K.A. De Jong, "A Cooperative Coevolutionary Approach to Function Optimization," in *Proc. of Parallel Problem Solving from Nature 3*, Springer-Verlag, Berlin, 1994.

22. Pfeifer, R., 1996. "Building "Fungus Eaters": design principles of autonomous agents," in *From Animals to Animats 4: Proc. of the Fourth International Conference on Simulation of Adaptive Behavior*, MIT Press, Bradford Books, Cambridge, MA, 1996.

23. I. Rechenberg, *Evolutionsstrategie*, Frommann-Holzboog: Stuttgart, 1973.

24. R. Salomon, "Performance Degradation of Genetic Algorithms under Coordinate Rotation," in *Proc. of the Fifth Annual Conference on Evolutionary Programming V*, MIT Press, Cambridge, MA, 1996.

25. R. Salomon, "Implicit Independence Assumptions; a Notorious Problem for Genetic Algorithms," in *Proc. of the International ICSC Symposia on Intelligent Industrial Automation and on Soft Computing (SOCO'96)*, ICSC Academic Press, Canada/Switzerland, 1996.

26. R. Salomon, "Increasing Adaptivity through Evolution Strategies," in *From Animals to Animats 4: Proc. of the Fourth International Conference on Simulation of Adaptive Behavior*, MIT Press, Bradford Books, Cambridge, MA, 1996.

27. R. Salomon, "Reevaluating Genetic Algorithm Performance under Coordinate Rotation of Benchmark Functions; A survey of some theoretical and practical aspects of genetic algorithms," in *BioSystems*, vol. 39, no. 3, pp. 263-278, 1996.

28. H.-P. Schwefel. *Evolution and Optimum Seeking*, John Wiley and Sons: Inc, New York, 1995.

29. K. Stokbro, D.K. Umberger, and J.A. Hertz, "Exploiting Neurons with Localized Receptive Fields to Learn Chaos," in *Complex Systems*, vol. 4, pp. 603-622, 1990.

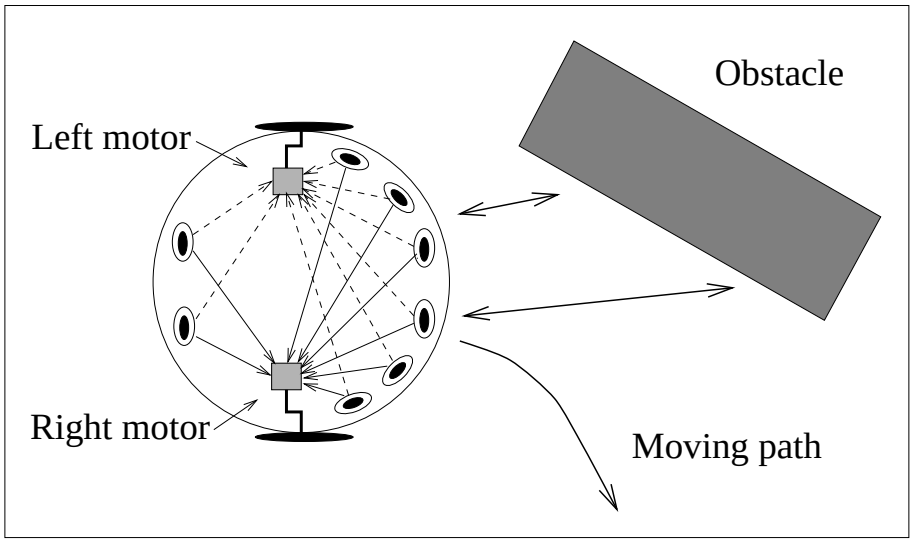

Figure 1. A control architecture inspired by a Braitenberg type-3C vehicle. The sensory information controls the motors via inhibitory and excitatory connections.

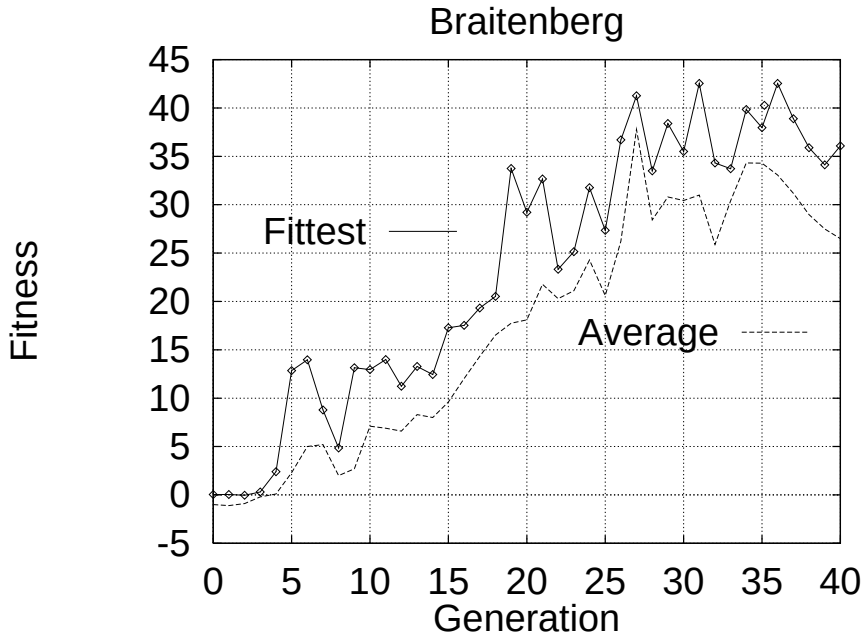

Figure 2. A typical run of the evolution of a *constrained* Braitenberg vehicle. The upper graph represents the best individual whereas the lower graph prepresents the average of the whole population.

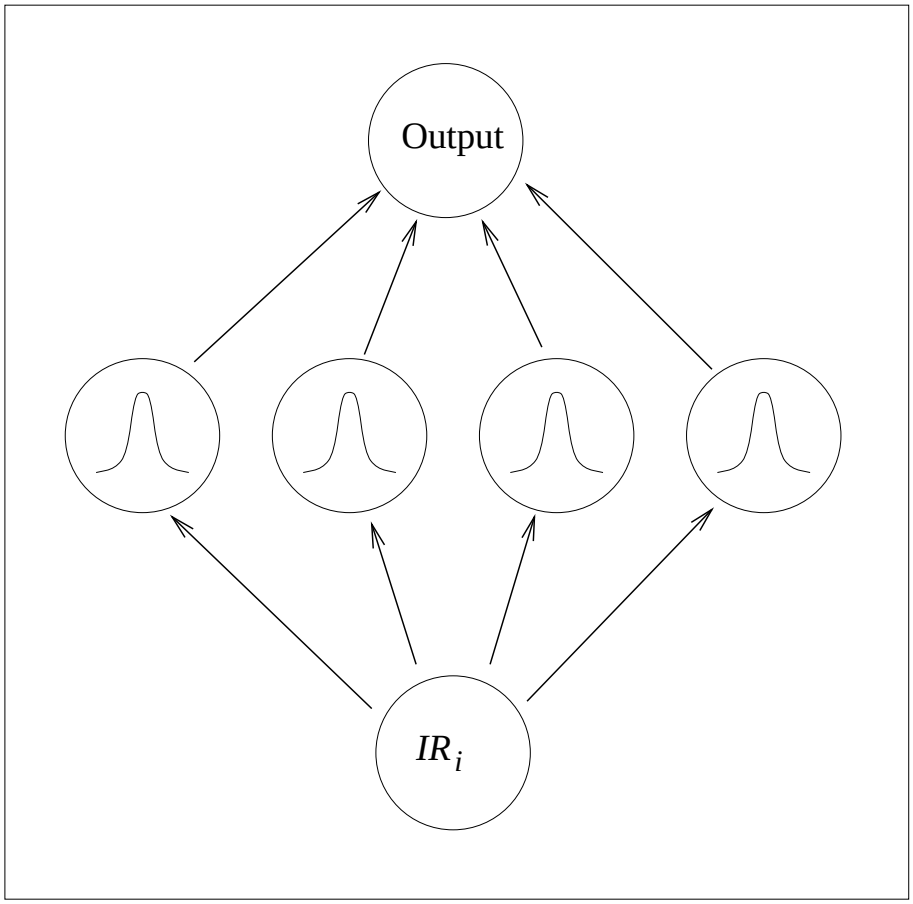

Figure 3. A one-dimensional receptive field.

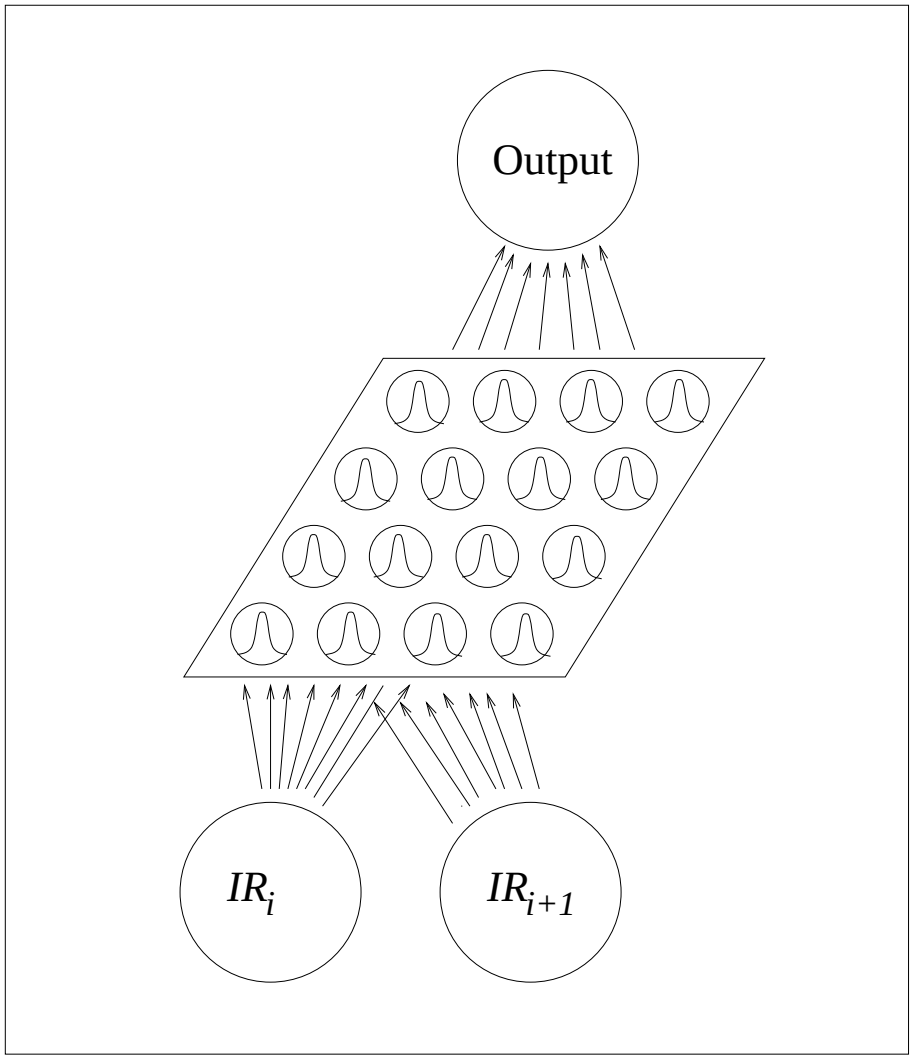

Figure 4. A two-dimensional receptive field.

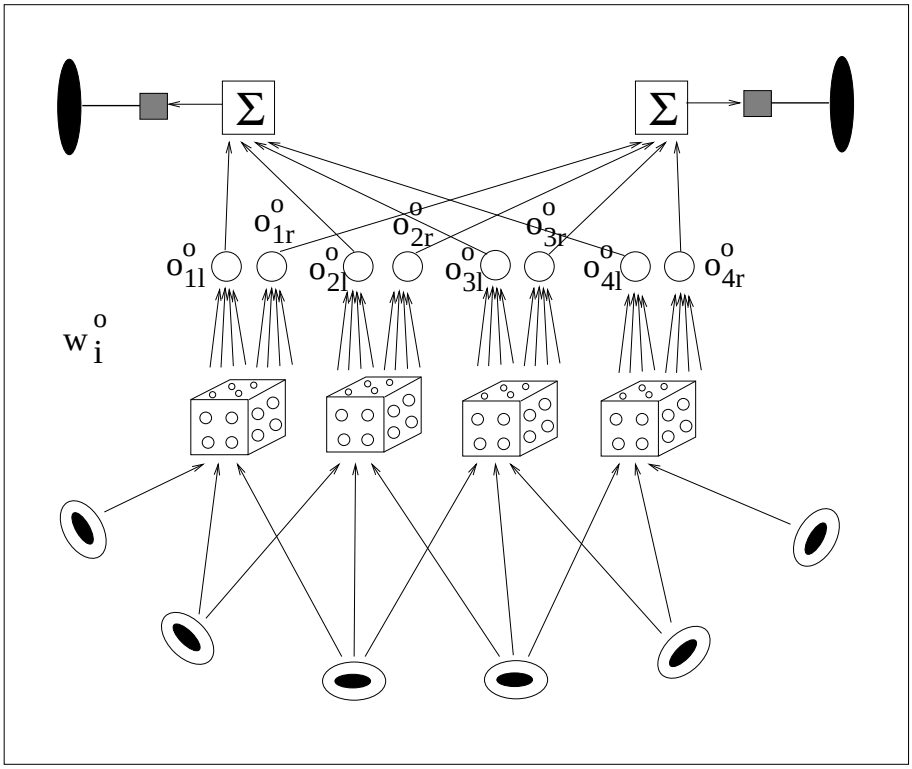

Figure 5. A complete architecture with three-dimensional receptive fields.

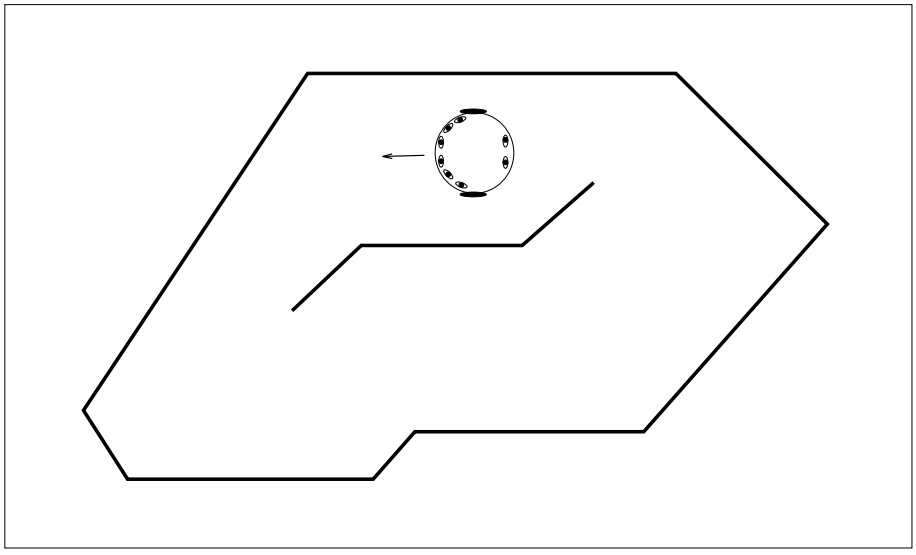

Figure 6. The arena of approximate size 60x45 cm. The indicated position (facing left) is the starting point for fitness evaluation.

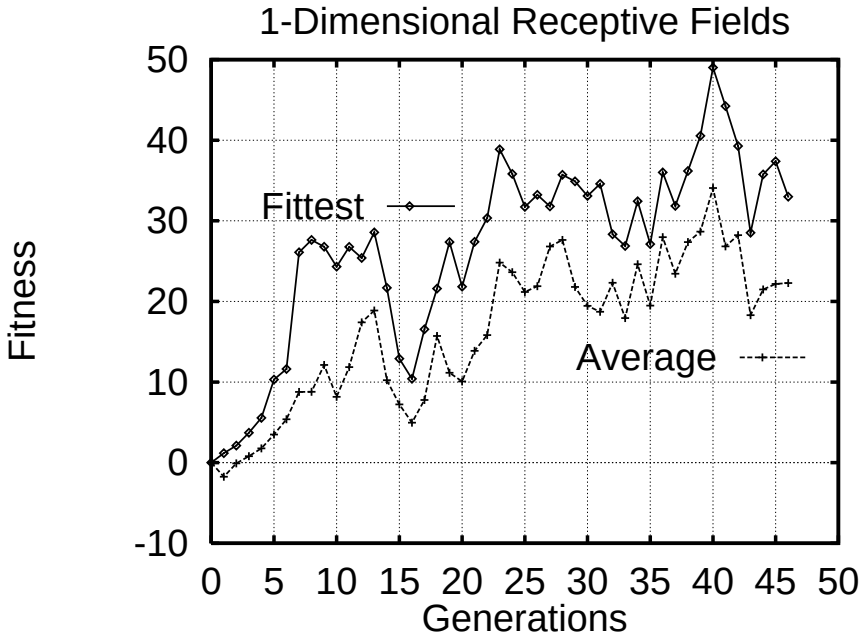

Figure 7. A typical run of the evolution of a constrained, one-dimensional receptive-field controller. The upper graph represents the best individual, whereas the lower graph prepresents the population average.

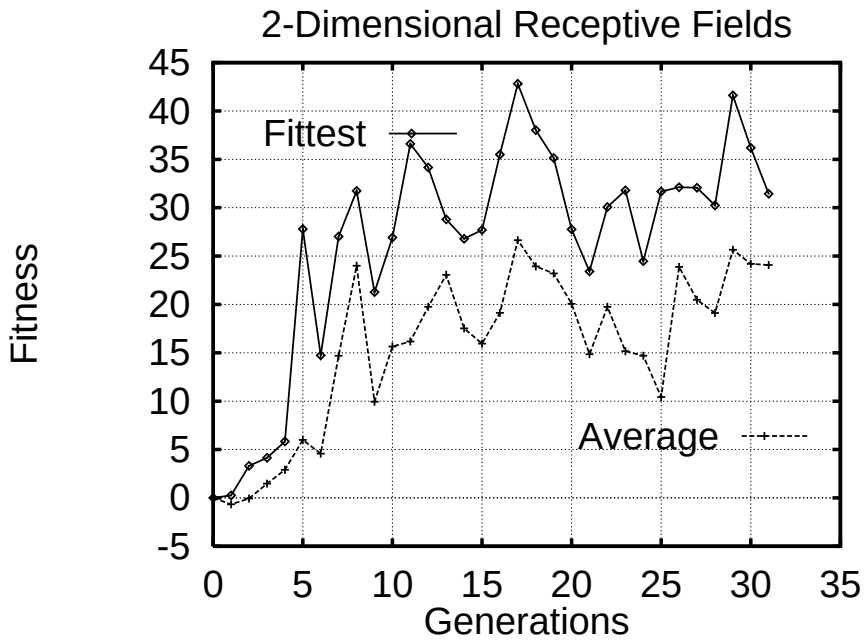

Figure 8. A typical run of the evolution of a constrained, two-dimensional receptive-field controller. The upper graph represents the best individual, whereas the lower graph prepresents the population average.

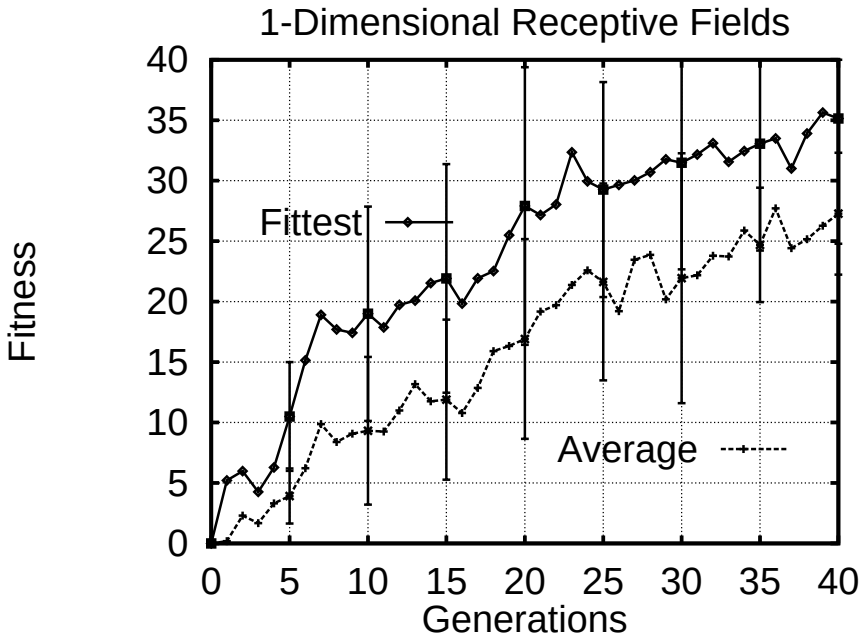

Figure 9. Averaged evolution of a constrained, 1-dimensional receptive-field controller including standard deviation.

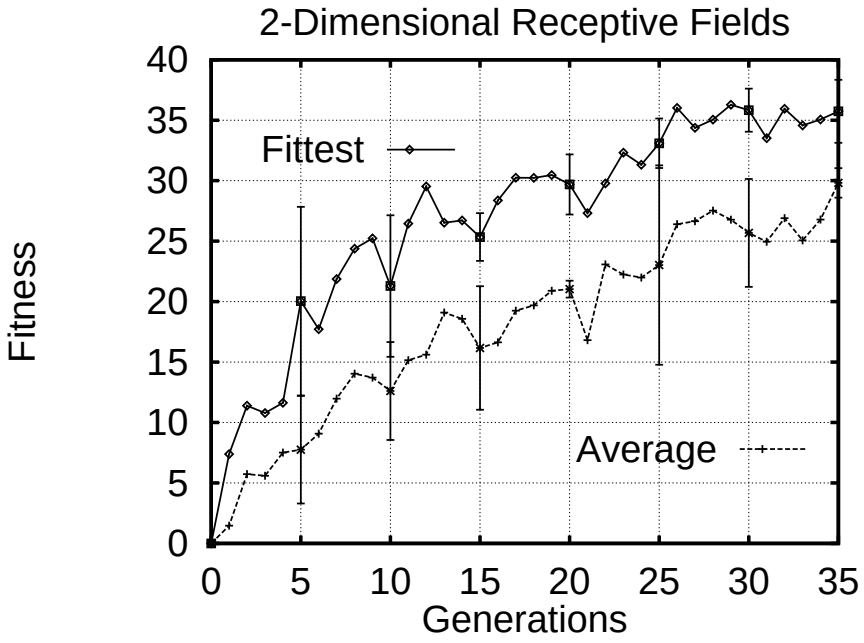

Figure 10. Averaged evolution of a constrained, 2-dimensional receptive-field controller including standard deviation.

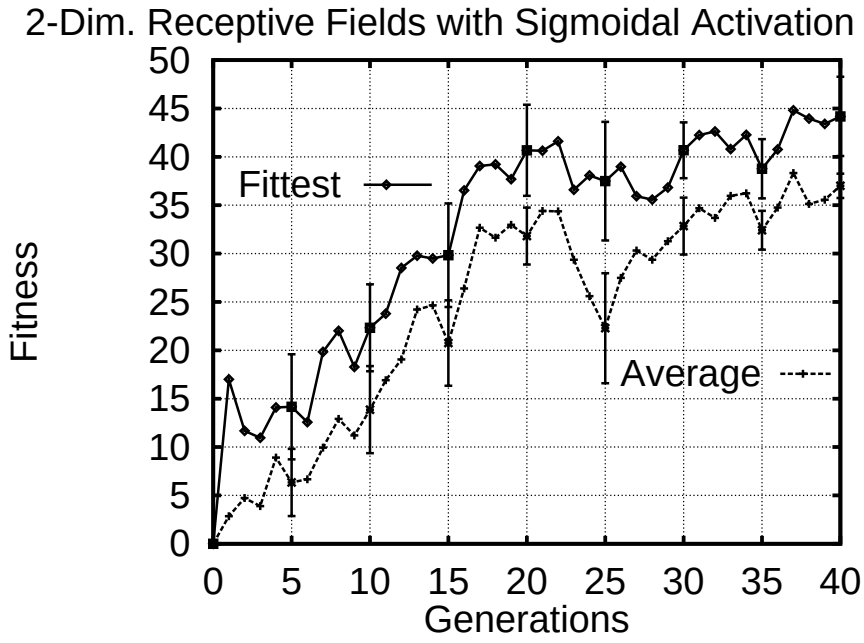

Figure 11. Average evolution of a constrained receptive-field controller with sigmoidal motor activation. The upper graph represents the average of the best individual whereas the lower graph prepresents the average of the population average.

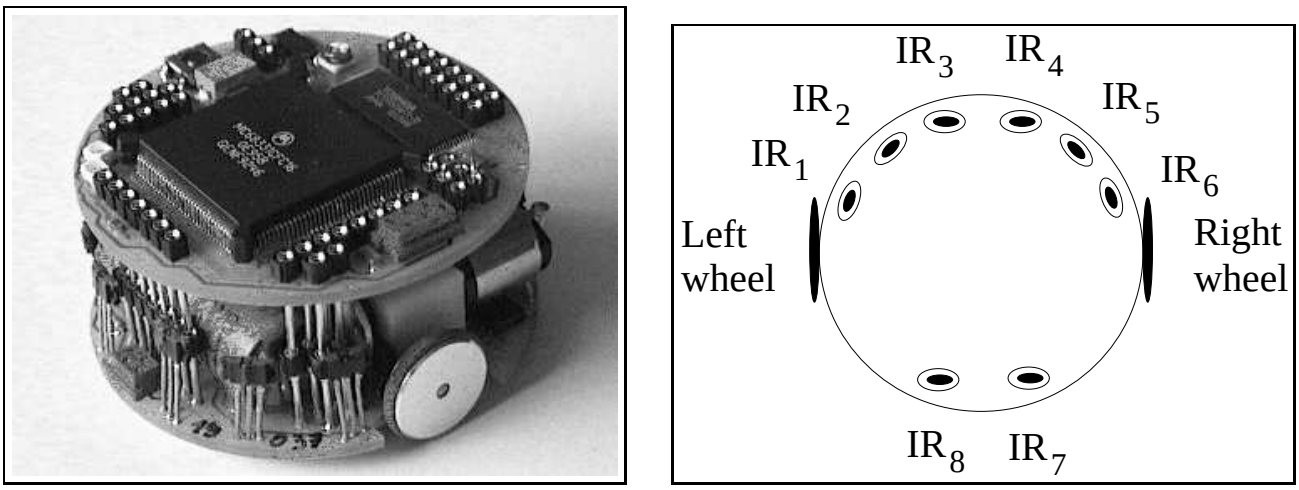

Figure 12. The Khepera robot. The right part shows the approximate location of the infrared sensors $IR_1 \ldots IR_8$.

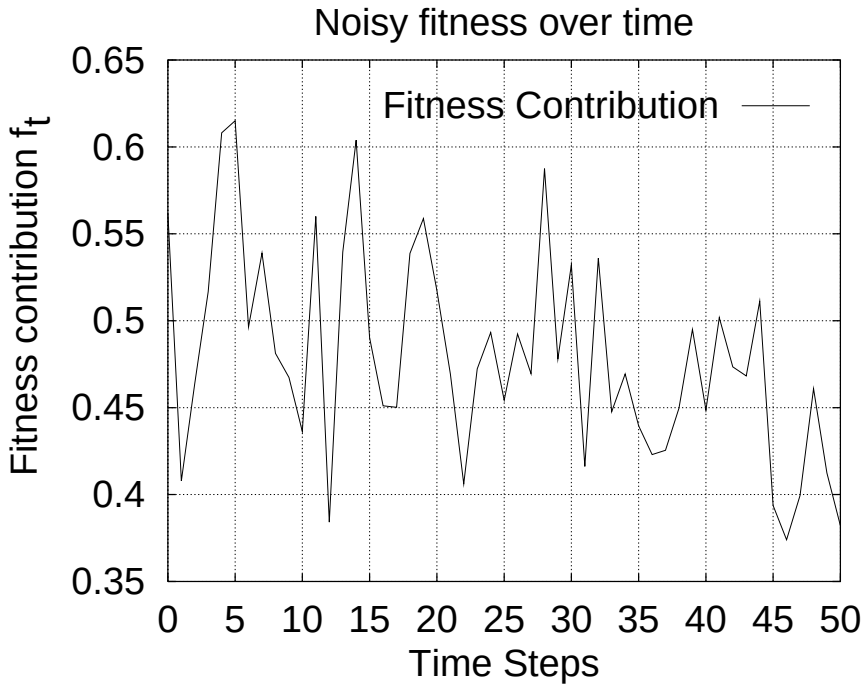

Figure 13. The noisy sensor readings cause dynamically changing fitness contributions f_t even under constant environmental conditions.

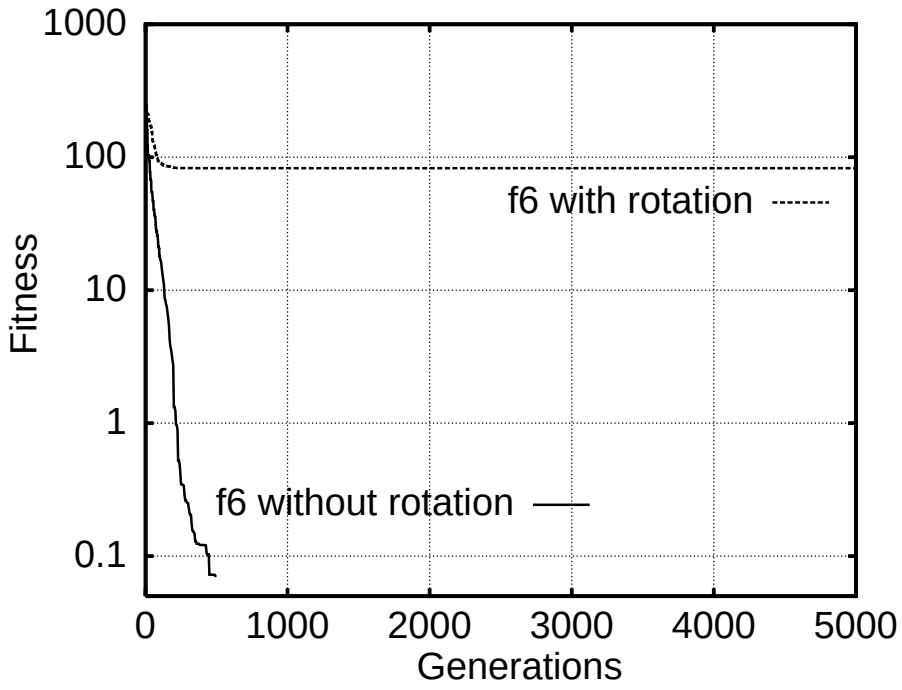

Figure 14. BGA's performance on Rastrigin's function f_6 with and without a rotation.

List of Figure Captions

Figure 1: A control architecture inspired by a Braitenberg type-3C vehicle. The sensory information controls the motors via inhibitory and excitatory connections.

Figure 2: A typical run of the evolution of a *constrained* Braitenberg vehicle. The upper graph represents the best individual whereas the lower graph prepresents the average of the whole population.

Figure 3: A one-dimensional receptive field.

Figure 4: A two-dimensional receptive field.

Figure 5: A complete architecture with three-dimensional receptive fields.

Figure 6: The arena of approximate size 60x45 cm. The indicated position (facing left) is the starting point for fitness evaluation.

Figure 7: A typical run of the evolution of a constrained, one-dimensional receptive-field controller. The upper graph represents the best individual, whereas the lower graph prepresents the population average.

Figure 8: A typical run of the evolution of a constrained, two-dimensional receptive-field controller. The upper graph represents the best individual, whereas the lower graph prepresents the population average.

Figure 9: Averaged evolution of a constrained, 1-dimensional receptive-field controller including standard deviation.

Figure 10: Averaged evolution of a constrained, 2-dimensional receptive-field controller including standard deviation.

Figure 11: Average evolution of a constrained receptive-field controller with sigmoidal motor activation. The upper graph represents the average of the best individual whereas the lower graph prepresents the average of the population average.

Figure 12: The Khepera robot. The right part shows the approximate location of the infrared sensors $\mathrm{IR}_1 \ldots \mathrm{IR}_8$.

Figure 13: The noisy sensor readings cause dynamically changing fitness contributions f_t even under constant environmental conditions.

Figure 14: BGA's performance on Rastrigin's function f_6 with and without a rotation.

Preferred address of contact author

Ralf Salomon
Department of Information Technology
University of Zurich
Winterthurerstrasse 190
8057 Zurich, Switzerland
email: salomon@ifi.unizh.ch
Phone: +41-1-635 45 77; FAX: +41-1-635 68 09