The Force Model: Concept, Behavior, Interpretation

Ralf Salomon[*]
[*] Department of Electrical Engineering and Information Technology
University of Rostock
18051 Rostock, Germany
ralf.salomon@etechnik.uni-rostock.de

Abstract

Most experiments in research on autonomous agents and mobile robots are performed either in simulation or on robots with static physical properties; evolvable hardware is hardly ever used. One of the very rare exceptions is the *eyebot* on which Lichtensteiger and Eggenberger have evolved simplified insect eyes (L. Lichtensteiger and P. Eggenberger, 1999). Even though substantially improved, the evolutionary models currently applied still lack both scalability and noise-resistance. To tackle these problems, this paper proposes a biologically-inspired force model for this class of real-world applications. The simulation results clearly indicate that this model provides a significant improvement over existing limitations. Furthermore, this paper argues that the force model is of more general utility.

1. Introduction

In its long history, (classical) artificial intelligence has developed its strengths in areas, such as playing games, developing game theory, automatic theorem provers, etc. Most of this research focuses on algorithmic questions that are more or less bound to a formal framework. Since the beginning of the 90ies, however, a new research area has emerged, for which Brooks has coined the term new artificial intelligence (new AI). New AI aims at understanding (natural) intelligence and its underlying mechanisms by building systems that exhibit "intelligent" behavior ??? (R. A. Brooks, 1991a, 1991b, R. Pfeifer and C. Scheier, 1999). These systems are often realized as mobile robots, which are supposed to operate in dynamically changing, partially unknown environments without any human control (they are thus attributed *autonomous*).

New AI prefers a synthetic approach, i.e., understanding by building. In order to reach its research goals, new AI draws a significant amount of inspiration from natural systems. It therefore often investigates (biological) hypotheses and aims at validating them in simulation or on particularly-designed robots. For obvious reasons, most validations are done in simulation (see, for example, the conference series *Simulation of Adaptive Behavior*), but some are indeed done on physical robots (L. Lichtensteiger and P. Eggenberger, 1999, R. Salomon, 1996, R. Salomon and L. Lichtensteiger, 2000). In most of these experiments, the robot's morphology, the positions of its sensors, etc., are predetermined, and adaptation concerns mostly the robot's controller, which is given in software in virtually all cases; *evolvable hardware* is still used only in exceptional cases (cf. (D. Keymeulen, M. Iwata, Y. Kuniyoshi, and T. Higuchi, 1999)).

Section 2. discusses the *eyebot*, which has been published in the recent literature (L. Lichtensteiger and P. Eggenberger, 1999) and is a nice example of a robot with evolvable hardware components. Lichtensteiger and Eggenberger used evolutionary algorithms to evolve the sensor distribution of an insect eye. As has been mentioned earlier, autonomous agents are supposed to freely move around and should not collide with obstacles. Therefore, Section 2. also explains how a suitable sensor distribution can be used to estimate the lateral distance to an obstacle by means of a very simple neural network, in which all connections have *equivalent* weights.

Unfortunately, evolution in hardware suffers from immense time requirements. On the eyebot, for example, one single fitness evaluation takes about one minute. Thus, if an evolutionary algorithm would consider a population of about 60 individuals, which are mostly considered not many, the execution of one single generation would already take an hour of experimentation time. In this setup, experimentation time is a very limited resource. Consequently, as Section 3. summarizes, subsequent research accelerated this application by developing different coding schemes (R. Salomon and L. Lichtensteiger, 2000). Despite the achievable performance improvements, the scalability remained strongly limited, which allows for the evolution of insect eyes with only a few receptors for practical reasons. Furthermore, the consideration of noise, which is omnipresent in such hardware setups, imposes severe problems; in many cases, the evolutionary algorithms converge at poor solutions.

In order to tackle the problems discussed above, Section 4. proposes a new, biologically-inspired coding approach, called the *force model*. In essence, this model can be considered a reformulation of the original task. Due to its very nature, this force model is not limited to the evolution of an inset eye, but may be transferred to other real-world applications. Section 5. summarizes the experimental setup, Section 6. then

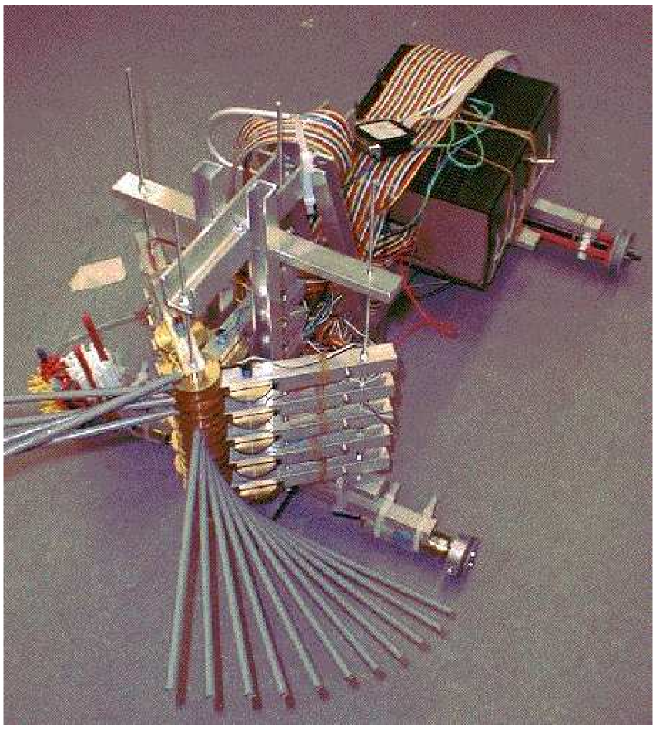


Figure 1: The "eyebot" consists of a chassis, an on-board controller, and sixteen independently-controllable facet units (see Fig. 2), which are all mounted on a common vertical axis.

presents some results, which indeed indicate a significant reduction of the problem's complexity, which offers the evolution of more complex systems. Section 7. tries to analyses the mechanisms responsible for the obtained performance improvements. Section 8. finally concludes with a brief discussion.

2. Background: The Eyebot

This section summarizes previous research and includes the description of the robot, its eye, the neural network controller, as well as the motion parallax phenomenon.

2.1 The Eyebot

Inspired by biological evidence (T. S. Collett, 1978, N. Franceschini, J. Pichon, and C. Blanes, 1992, G. A. Horridge, 1978), Lichtensteiger and Eggenberger (1999) constructed the *eyebot* (Fig. 1) to model the eye of an insect, such as the house fly. It consists of a chassis, an on-board controller, and sixteen independently-controllable facet units, which are all mounted on a common vertical axis. A facet unit (Fig. 2) basically consists of the sensor, a thin tube, two cog-wheels, a motor, and a potentiometer. By means of the cog-wheels, the motor can position the facet within a range of about 200 degrees, and the potentiometer provides feedback about its actual position, i.e., its angle α_i. The thin opaque tube is used to reduce the sensor's aperture to about two degrees.

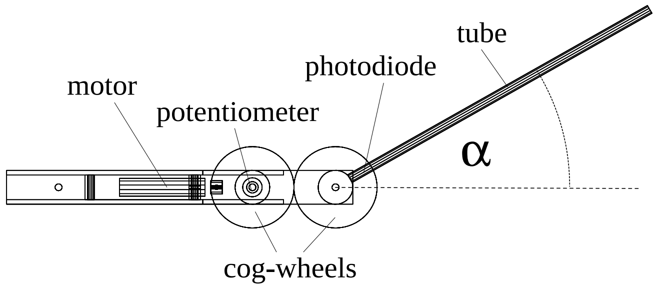


Figure 2: The robot's facet basically consists of the sensor (the photo diode), a thin tube, two cog-wheels, a motor, and a potentiometer. By means of the cog-wheels, the motor can position the facet within a range of about 200 degrees, and the potentiometer provides feedback about its actual position.

These tubes are the primitive equivalent to the biological ommatidia (T. S. Collett, 1978, N. Franceschini, J. Pichon, and C. Blanes, 1992, G. A. Horridge, 1978). It should be noted that such a low-cost construction is subject to several imprecisions and tolerances, which might be sensed as noise during operation.

2.2 Motion Parallax: A Mathematical Description

Figure 3 sketches how a phenomenon, also known as motion parallax, can be utilized to avoid obstacles. Let us assume that the compound eye, presented as the observer R, is at a fixed position. If the obstacle X is moving at constant speed v, the observer views the obstacle under different angles α_1, α_2, and α_3 at different time steps t_1, t_2, and t_3. Let r denote the vector from observer R to obstacle X and let v_t denote the component of v that is perpendicular to vector r. For the distance d of closest approach, the following relation holds

$$d = r \sin\alpha \; . \tag{1}$$

Since $v_t = v \sin\alpha$, the angular velocity $\omega = \dot{\alpha}$ with which the image of X moves through the visual field of the observer is

$$\omega = \frac{v_t}{r} = \frac{v \sin\alpha}{r} \; . \tag{2}$$

If the agent can estimate the velocity v and can measure both α and ω, it can calculate its distance r to the obstacle at any time. Solving eq. (2) for r and substituting into eq. (1) leads to

$$d = \frac{v}{\omega} \sin^2\alpha \; . \tag{3}$$

Let us assume that the agent uses some sensors each of which can detect the obstacle if it appears under a particular angle α. The agent can then estimate the angular velocity $\omega = d\alpha/dt \approx \Delta\alpha/\Delta t$ by the change of α per time interval:

$$d = \frac{v}{(d\alpha/dt)} \sin^2\alpha \approx v \frac{\Delta t}{\Delta\alpha} \sin^2\alpha \; . \tag{4}$$

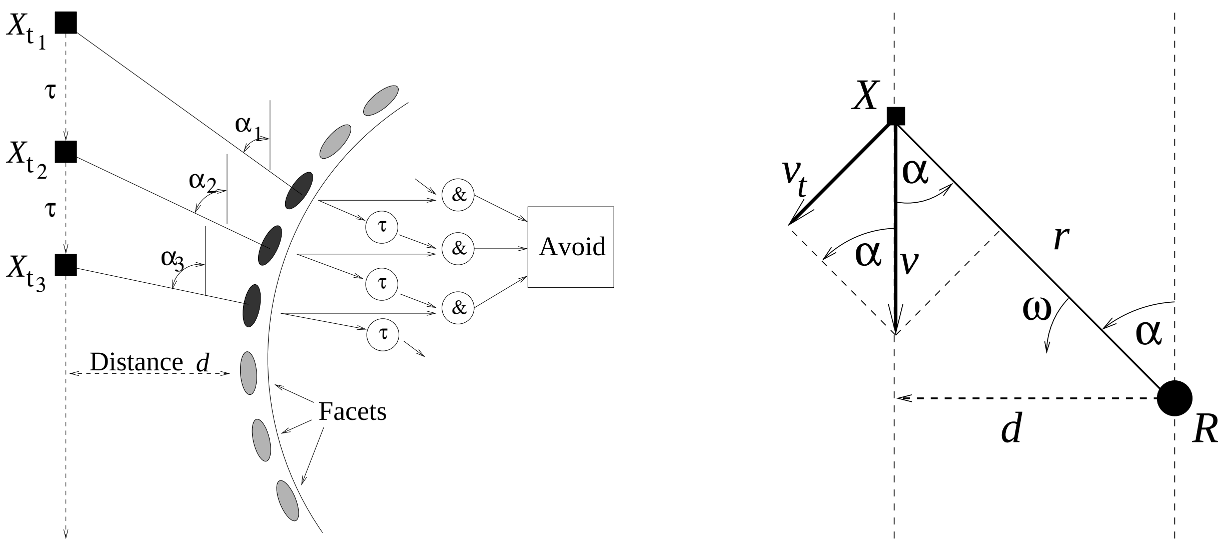


Figure 3: Left: due to their small aperture of about two degrees, the insect eye's facets recognize a moving object at different times $t_1, \dots t_3$. With some units that maintain their activity during a time period τ after stimulation and output units that perform an and-operation over neighboring input pairs, a neural network is able to detect too small a distance of closest approach to an obstacle. Right: the angular velocity ω under which an obstacle X is seen equals $\omega = v_t/r = v \sin\alpha/r$ and depends on the actual angle α leading to a distance of closest approach $d = (v/\omega)\sin^2\alpha$.

Similar to biological systems, an agent with s facets can estimate $\omega \approx \Delta\alpha/\Delta t$ by utilizing a simple neural network, which consists of s input units u^I and s-1 output units u^O. As can be seen in Fig. 3, each output unit u_i^O is connected to two input units u_i^I and u_{i+1}^I each of which is in turn connected to one facet with all connections being topology preserving, i.e., neighboring facets signal to neighboring inputs, which in turn connect to the same output unit. Furthermore, each input unit u_i^I has an associated time constant τ_i during which the unit remains active after it has been triggered by an appropriate input.

Each triple u_i^I, u_{i+1}^I, and u_i^O constitutes a motion sensor. An input unit u_i^I is activated by the appearance of a sufficiently-high "dark-to-bright" stimulus. Then, this unit remains active during the decay time τ_i. If also the neighboring input unit u_{i+1}^I becomes active during this time interval, the output unit u_i^O is triggered (due to its "and" operation). If however, the stimulus moves too slowly, the first input neuron u_i^I becomes inactive and the output unit u_i^O is not triggered. Depending on its relative velocity v, the agent may trigger an appropriate avoidance action if the sum of active output units exceeds a critical threshold, which can be, for example, dependent on v (see also eq. (4)). For an agent to be successful, it is essential to avoid obstacles only if necessary, since an agent that constantly avoids obstacles is rather useless. Therefore the parameter τ_i determines a critical speed between two neighboring sensors.

An agent is in principle able to calculate the distance d of closest approach by utilizing only two facets. However, this calculation would be limited to a particular angle of its visual field, which would be too restrictive for real-world mobile robots. By using a compound eye, the agent is able to anticipate potential collisions regardless of the angle under which the object is seen. That is, two compound eyes would provide an agent with an almost 360° view.

2.3 Evolving A Suitable Sensor Distribution

Equation (4) for calculating the latera distance d is highly non-linear in α. If all network delays τ are assumed equivalent, then the time it takes the obstacle to be recognized by two neighboring facets must be τ as well (see also Fig. 3). This immediately implies that the distance between two intersections of the facets and the moving path must be equivalent as well. This, in turn, suggests the following fitness function

$$f = \sum_{i=1}^{s-1} (t_i - \tau)^2 . \qquad (5)$$

As already mentioned in the introduction, this paper focuses on the development of a potentially non-uniform sensor distribution (i.e., the α_i's) (L. Lichtensteiger and P. Eggenberger, 1999), which constitutes the eye's morphology, for a simple, prespecified network, rather than developing a complicated neural network for a – perhaps inadequate – sensory system. From Eq. (4), it can be easily derived that the target distribution is $d\alpha/dt \sim \sin^{-1}\alpha$. It should be furthermore noted that all benchmark tests are done in simulation but that particular results are validated by physical robot experiments.

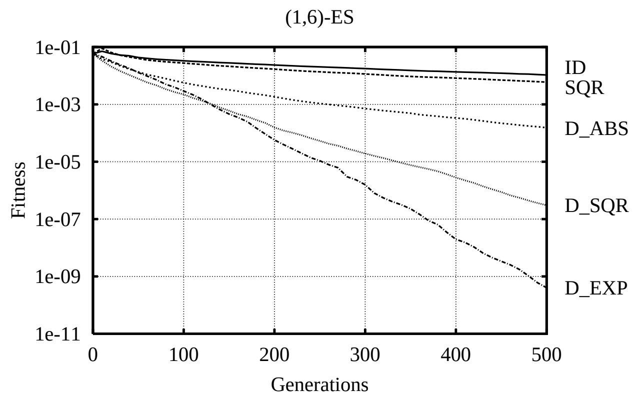


Figure 4: The performance of the (1,6)-evolution strategy significantly depends on the chosen coding scheme.

3. Previous Research and Problem Description

Most classical optimization procedures would try to optimize the given objective (Eq. (5)) by directly applying mathematical operators to the problem-specific variables, which are the angles α_i in the problem at hand. Examples for such operators are the gradient or even the Hessian-matrix.

Rather than applying mathematical (optimization) operators to only one object, most evolutionary algorithms consider a population of different objects and apply simple (random) variation operators, such as mutation and recombination/crossover. It is widely believed that this population-based approach has certain advantages, such as resistance to noise, not getting trapped in local optima, not being biased by the designer's preferences, and so forth.

It might be interested to note that also most evolutionary algorithms apply their variation operators directly to the individuals' problem-specific variables. However, this is *not mandatory*, because the evolutionary context distinguishes between genotype (the parameters x_i to be considered) and the object's phenotype (i.e., the angles α_i in the problem at hand). Evolutionary algorithms therefore employ am appropriate mapping function; but surprisingly many research attempts consider only the identity function.

Even though successful, preliminary experiments (L. Lichtensteiger and P. Eggenberger, 1999) (as mentioned above) indicated slow convergence when using the identity mapping functions. Some other previous research (R. Salomon and L. Lichtensteiger, 2000) investigated the efficacy of the following mapping functions (coding schemes): ID: $\alpha_i = x_i$; SQR: $\alpha_i = x_i^2$; D_ABS: $\alpha_{i+1} = \alpha_i + |x_i|$; D_SQR: $\alpha_{i+1} = \alpha_i + x_i^2$; and D_EXP: and $\alpha_{i+1} = \alpha_i + c * \exp(x_i)$, with c denoting an arbitrary scaling constant, which was set to $c = 1.8$ for historical reasons. All coding schemes assume that the angles α_i are numbered from front to side (see Fig. 3).

Figure 4 shows that the choice of the coding scheme significantly influences the efficacy of a simple (1,6)-ES (please, see Section 5. for the experimental setup). Further results indicate that evolution strategies converge significantly faster than genetic algorithms on this particular problem (R. Salomon and L. Lichtensteiger, 2000).

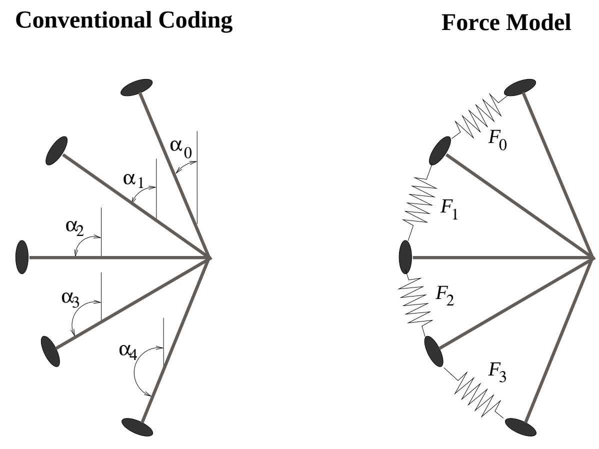


Figure 5: Both figures illustrate an artificial insect eye with five facets. A conventional model (left-hand-side) would directly or indirectly focus on the various angles α_i. By contrast, the force model (right-hand-side) interprets the objective variables as forces F_i between two neighboring facets, and derives the various angles as $\alpha_{i+1} = \alpha_i + F_i(\alpha_n - \alpha_0)/(\sum_{j=0}^{n-1} F_j)$.

Despite all achieved performance improvements, the following two problems remain:

1. The scalability is still unsatisfactory for an increased number of facets s.
2. All coding schemes reacted very sensitive to a noisy fitness evaluation, which is the normal case in a real-world setup.

The next section proposes a new model, called the *force model*, to tackle these problems by new different approach.

4. The Force Model

The various coding schemes discussed so far lack one common problem. A modification of one of the very first angles moves all other facets by that amount. This can result in drastic fitness changes under certain circumstances; unfortunately, these models do not provide a gradually decreasing shift. From a theoretical point of view, it might be better, if a shift of x-degrees of one facet would be distributed in small portions $x/(n-1)$ over the remaining $n-1$ facets. This idea has led to the following force model:

- The eye consists of $n+1$ facets from which the first and last are kept in fixed positions, i.e., α_0 and α_n are fixed, by the agents body.
- An individual has n objective variables which are directly or indirectly considered as forces F_i between the two

facets i and $i+1$. Due to these n forces F_i, the remaining $n-2$ facets evolve their final positions.

- The individual angels α_i are not directly coded. Instead, the model *virtually simulates* springs between neighboring facets. These springs expand and/or shrink depending on some internal forces. Since all facets' tubes are of equal length, the angle α_i is directly proportional to the spring's extension. The force model adjusts the angle between two neighboring facets in proportion to the spring's internal force (and thus to its extension). The angular opening between two neighboring facets is then $\alpha_{i+1} - \alpha_i = F_i / \sum_j F_j$. Since the model assumes equal material constants, it is sufficient to only consider forces (instead of spring extensions). The force model derives all angles as follows:

$$\alpha_{i+1} = \alpha_i + F_i \frac{\alpha_n - \alpha_0}{\sum_{j=0}^{n-1} F_j} \qquad (6)$$

To repeat shortly: The force model employs $n+1$ facets from which two are kept at fixed positions. Between those $n+1$ facets, the model considers n forces with which the remaining $n-1$ facets evolve their final positions. In other words, this model has one more genotype parameter than phenotype attributes. In the remainder of this paper, the number of free facets is denoted by $s = n-1$. (Please remember that the model employs n+1 facets from which two are hold at fixed positions.)

The force model draws its inspiration from some biological observations. Consider a bunch of cells. The insertion of another cell (or a small number of cells) at some particular place would have rather "strong" effect its vicinity but a rather small global one. The force model behaves similarly. A particular force F_i twice as strong as before would almost double the angle between facets i and $i+1$, and would approximately decrease all other angles by the same amount but divided by n.

5. Methods

The term evolutionary algorithms refers to a class of heuristic population-based search procedures that incorporate random variation and selection, and provide a framework that mainly consists of genetic algorithms (D. E. Goldberg, 1989), evolutionary programming (L. J. Fogel, 1962, D. B. Fogel, 1995), and evolution strategies (I. Rechenberg, 1973, 1994, H.-P. Schwefel, 1995).

Even though all evolutionary algorithm have their own peculiarities, they share a lot of common features. All evolutionary algorithms maintain a population of μ individuals, also called parents. In each generation g, an evolutionary algorithm generates λ offspring by copying randomly selected parents and applying variation operators, such as mutation and recombination. It then assigns a fitness value (defined by a fitness or objective function) to each offspring. Depending on their fitness, each offspring is given a specific survival probability. For a good overview of these algorithms, the interested reader is referred to (T. Bäck, U. Hammel, and H.-P. Schwefel, 1997).

Since in the compound eye all angles α_i are real-valued parameters, this paper employs evolution strategies (I. Rechenberg, 1973, 1994, H.-P. Schwefel, 1995) and the breeder genetic algorithm (H. Mühlenbein and D. Schlierkamp-Voosen, 1993).

5.1 *Evolution Strategies*

In their simplest form, evolution strategies maintain one global step size σ for each individual, and they typically apply mutations to all n parameters $x_i, 1 \le i \le n$, i.e., $p_m = 1$, as follows

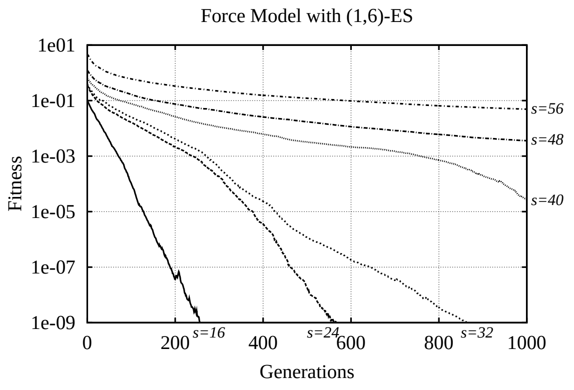
$$x_i \leftarrow x_i + \sigma N(0,1) \ , \qquad (7)$$

with $N(0,1)$ denoting normally-distributed random numbers with an expectation value of 0 and a standard deviation of 1. Each offspring inherits the step size from its parent, and prior to mutation, the inherited step size is modified by multiplication with a lognormally-distributed random number[1] $\exp(N(0,1))$. In all experiments, the step size was initially set to $\sigma = 0.01$. This simple evolution strategy is denoted as (μ,λ)-ES, or (μ+λ)-ES. The first selection scheme indicates that the algorithm chooses the parents for the next generation *only* from the offspring, whereas the latter selection scheme selects from the union of the offspring *and* previous parents, i.e., μ-fold elitism. In addition, some evolution strategies also feature various recombination operators (see (T. Bäck, U. Hammel, and H.-P. Schwefel, 1997) for further details), such as discrete and intermediate recombination. More elaborate evolution strategies feature individual step sizes σ_i, one for each parameter x_i. Since these forms require relatively large population sizes of $\lambda \ge 200$ to work properly (H.-P. Schwefel, 1997), and since experimentation time is a severe constraint for the final real-world experiments, these forms are not considered here.

5.2 *The Breeder Genetic Algorithm*

The breeder genetic algorithm (H. Mühlenbein and D. Schlierkamp-Voosen, 1993), denoted as (μ,λ)-BGA for short, is a genetic algorithm variant that is especially tailored to continuous parameter optimization. The breeder genetic algorithm also encodes all parameters x_i as floating-point numbers, and implements mutation by adding or subtracting small random numbers. It normally applies a mutation to each parameter with probability $p_m = 1/n$. In addition, it features different crossover operators, such as discrete recombination, extended intermediate recombination, and extended line recombination (see (H. Mühlenbein and D. Schlierkamp-Voosen, 1993) for further details). It is recommended (H. Mühlenbein and D. Schlierkamp-Voosen, 1993) to use dis-

[1]Constant factors, such as 1.5, 1.0, and 1/1.5, might work as well; see also (I. Rechenberg, 1973).



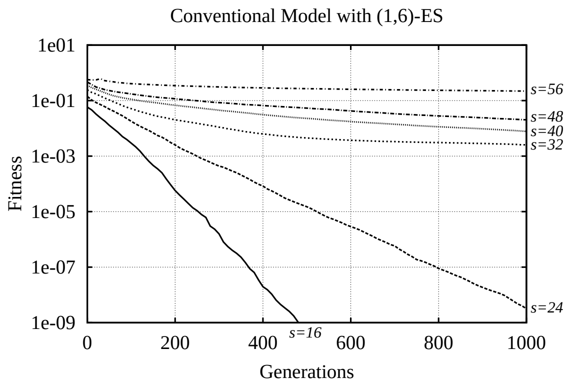
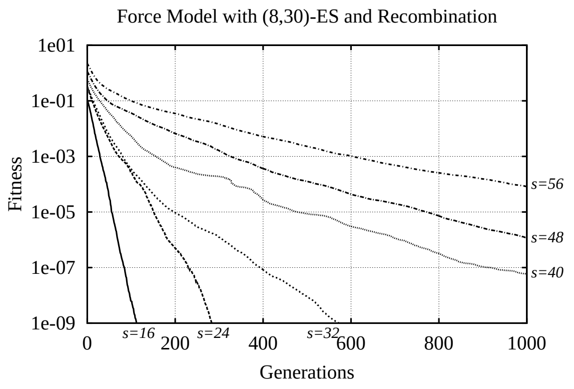


Figure 6: Performance of the (1,6)-ES for various numbers of facets s.



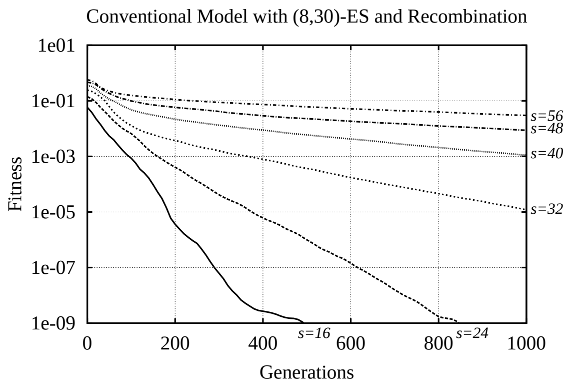


Figure 7: Performance of the (8,30)-ES and recombination for various numbers of facets s.

crete recombination with $p_r = 0.5$. The current breeder genetic algorithm's mutation operator (D. Schlierkamp-Voosen and H. Mühlenbein, 1994) is typically defined as

$$x_i \leftarrow x_i \pm A 2^{-ku} \; , \quad u \in [0,1) \; , \tag{8}$$

with "+" and "-" being selected with equal probability, A denoting the mutation range, k denoting a precision constant, and u being a uniformly-distributed random number. Unless otherwise stated, $A = 0.1$ and $k = 16$ were used in all experiments. Previously, discrete mutations were used (H. Mühlenbein and D. Schlierkamp-Voosen, 1993). Furthermore, the breeder genetic algorithm features simple elitism by preserving the best parent from the previous generation in case all offspring have worse fitness.

5.3 *The Fitness Function*

The fitness of a particular sensor distribution is given by the sum of the squared deviations to the time constant τ (τ is equal for all neurons in order to obtain the most simple network):

$$f = \sum_{i=0}^{s} (t_i - \tau)^2 \; . \tag{9}$$

When moving with speed v, the time interval t_i is given by the lateral distance between two neighboring sensors

$$t_i = \frac{d}{v} \left(\frac{1}{\tan(\alpha_i)} - \frac{1}{\tan(\alpha_{i+1})} \right) \; . \tag{10}$$

To allow the consideration of noisy fitness evaluations, this paper also considers the following fitness definition

$$f = \sum_{i=0}^{s} ((1 - xN(0,1))t_i - \tau)^2 \; , \tag{11}$$

with x denoting the Gaussian-distributed noise level.

Unless otherwise stated, $d/v = 1$ and $\tau = 0.15$ have been used in all experiments. With $\alpha_0 = 20$ degrees and s free facets, the system has s remaining free parameters $\alpha_1 \ldots \alpha_s$.

6. Results

Figures 6 to 9 compare some representative performance aspects of the two models under consideration. The figures on the left-hand-side alway refer to the force model, whereas the figures on the right-and-side always refer to the rather conventional α_i-coding scheme. All shown graphs are averages over 20 independent runs.

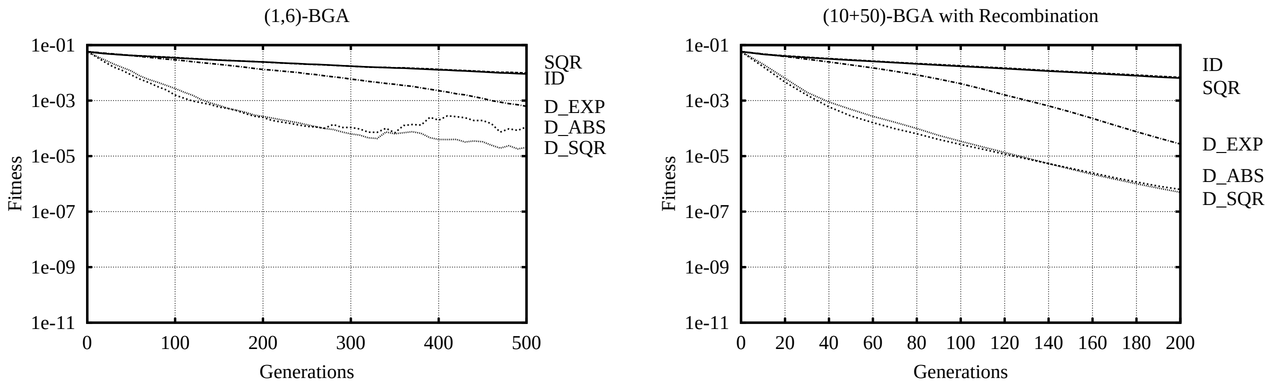


Figure 8: Performance of the (1,6)-BGA and the (10+50)-BGA for only s=16 facets and different coding schemes. Compare also to Fig. 4.

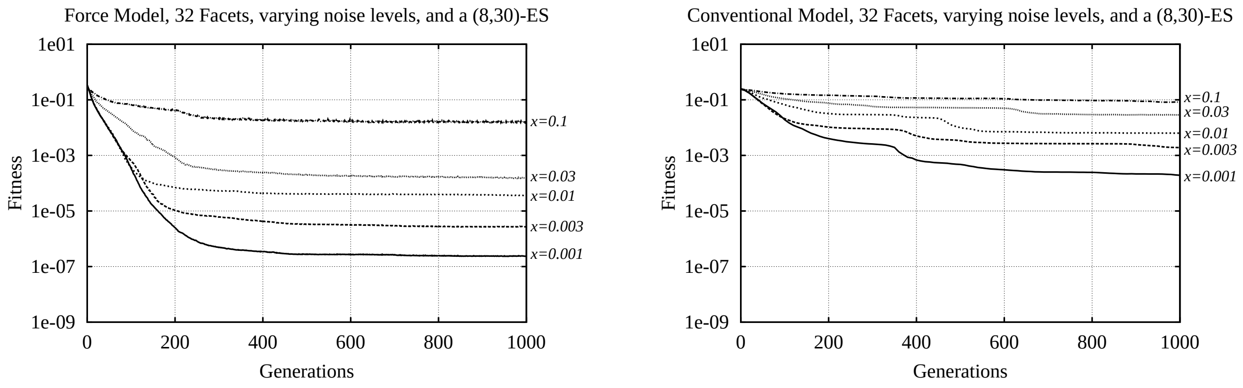


Figure 9: Performance of the (1,20)-ES for various noise levels x and $s = 32$ facets.

Figure 6 indicates that with the force model, the considered evolution strategies find precise solutions significantly faster; especially for larger numbers of facets s, the difference increases up to an order of magnitude. This gain is practically relevant, since it reduces the experimentation time of real-world experiments from an entire day to approximately three hours. As Fig. 7 shows, evolution strategies yield a similar behavior when changing their parametrization, i.e., μ and λ.

The results obtained by using the BGA are not very satisfying. As Fig. 8 shows, the results (in comparison to Fig. 4) for using only s=16 facets are already very poor. Since the performance progressively degrades when using significantly more facets, the presentation of additional results seem not worth it.

In addition to the presentation of some practically-relevant performance gains, this paper wants to emphasize on the force model's advantage in the case of a noisy fitness evaluation. Figure 9 clearly shows that the force model exhibits both a significantly higher rate of descent and a final accuracy orders of magnitude better than the conventional α_i-coding model. For example, with a noise level of 0.1% ($x = 0.001$), the force model achieves a final fitness value of about 2.4 10^{-7}, a value never reached by the conventional model; by contrast, the conventional model takes twice as long to end up more than three orders of magnitude worse.

At this point, two short notes should be made. First, the resulting sensor distribution is not affected by the proposed coding scheme, since the fitness functions (eq. (11)) has not been changed. Second, a generalization to other problems in this domain is straight forward; the particular design step is to account for give real-world physical constraints.

7. Interpretation

This section addresses the question of why the force model significantly improves the evolutionary process. The obtainable speed up is – in a sense – mainly due to the normalization given in eq. (6). The significant effect caused by this little change can be understood as follows: Let us assume, for the sake of simplicity, that the fitness function has only to parameters. In the conventional model, the evolutionary process has to evolve to a pair of very specific values. In a three-dimensional surface landscape, the optimum is represented by the lowest point of a two-dimensional parabola-like function.

Figure 10 depicts how the optimization process might be finding the *only* global optimum. It is evident that the successful optimization steps necessarily decrease as the opti-

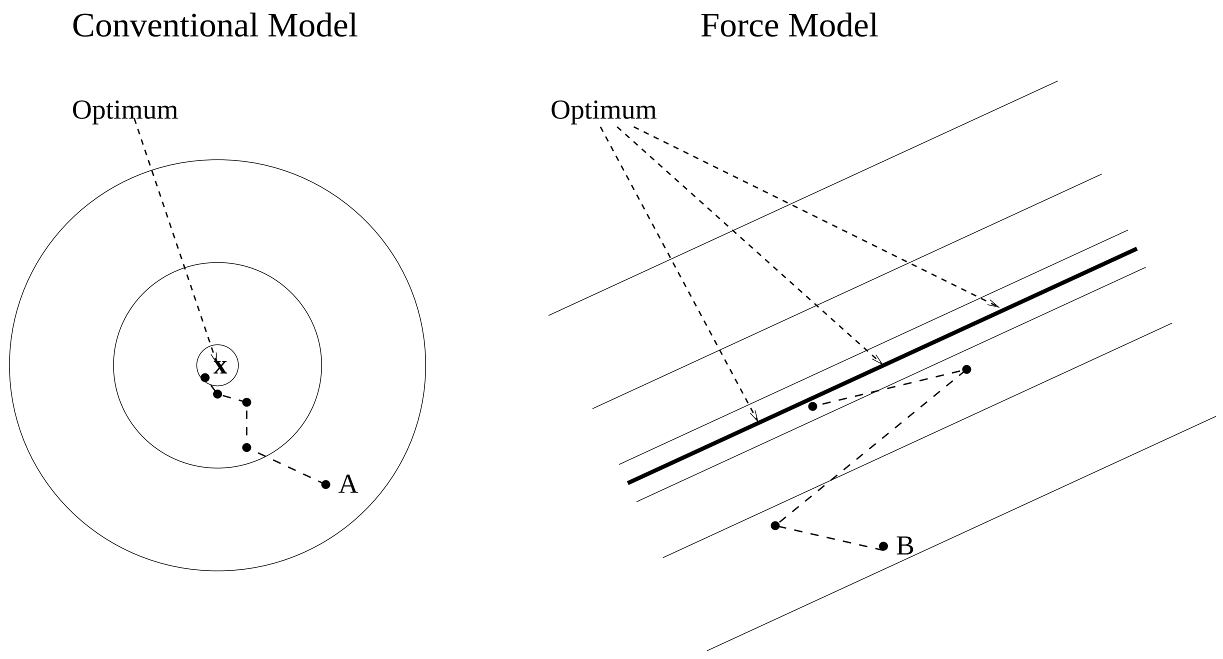


Figure 10: The fi gure on the left-hand-side shows the rather conventional model that has exactly one optimum. As this fi gure shows, the possible steps that yield a fi tness improvement are constantly decreasing as the optimization process progresses. The fi gure on the right-hand-side illustrates the situation of the force model. Rather than having *only* one optimum, the force model has infi nitely many that are all along a particular line. As can be seen, the optimization process has many more possibilities to achieve an improvement, and consequently, the optimization process might be much faster. "A" and "B" denote some comparable starting points.

mization process progresses. Hence, the optimization process requires an increasing amount of time.

The situation changes in case of the force model. Due to the normalization step, the evolutionary process does consider *only* the quotient of these two parameters. That is, instead of searching for two specific values x_1^o and x_2^o, the force model aims at finding $x_1/x_2 =$ opt. In this case, however, as Fig. 10 shows, the optimum changes from a single point to an entire line (the base of a valley) with infinitely many optimal relations $x_1/x_2 =$ opt. Hence, the force model's normalization step at least reduces the complexity. Apparently, this modification also makes the step size adaptation easier, and thus accelerates the optimization process. It furthermore seems as if the existence of infinitely many optima also positively affects the presence of noise.

8. Discussion

This paper has proposed the force model for the evolution of a simplified insect eye directly in hardware. Despite its acceleration, the force model behaves very resistant to external disturbances (noise), which are omnipresent in real-world experimental setups. Furthermore, the force model applies a normalization which reduces the number of parameters by one.

The achieved improvements are probably mainly due to the model's normalization. This normalization changes the fitness landscape's structure. If, for example, a two-dimensional instance of the conventional model contains a unique optimum at α_1^o, α_2^o, the normalized model contains infinite many optima at $\alpha_1^o = c \times \alpha_2^o$. Future research will be devoted to a thorough theoretical analysis.

References

T. B¨ack, U. Hammel, and H.-P. Schwefel (1997). Evolutionary Computation: Comments on the History and Current State. *IEEE Transactions on Evolutionary Computation,* 1(1):3-17.

R. A. Brooks (1991a). Intelligence Without Reason. *Proceedings of the 12th Intl. Conference on Artificial Intelligence (IJCAI-91),* Morgan Kaufmann, San Mateo, CA, 569-595.

R. A. Brooks (1991b). Intelligence without representation. *Artificial Intelligence,* 47:139-159.

T. S. Collett (1978). Peering – a locust behavior pattern for obtaining motion parallax. *Journal of Experimental Biology,* 76:237-241.

L. J. Fogel (1962). Autonomous Automata. *Industrial Research* 4:14-19.

D. B. Fogel (1995). *Evolutionary Computation: Toward a New Philosophy of Machine Learning Intelligence.*

N. Franceschini, J. Pichon, and C. Blanes (1992). From insect vision to robot vision. *Philosophical Transactions of the Royal Society of London B,* 337:283-294.

D. E. Goldberg (1989). *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, Reading, MA.

G. A. Horridge (1978). Insects which turn and look. *Endeavour* 1:7-17.

L. Lichtensteiger and P. Eggenberger (1999). Evolving the Morphology of a Compound Eye on a Robot. *Proceedings of the Third European Workshop on Advanced Mobile Robots (Eurobot '99)*. IEEE Piscataway, NJ, pp. 127-134.

D. Keymeulen, M. Iwata, Y. Kuniyoshi, and T. Higuchi (1999). Online Evolution for a Self-Adapting Robotic Navigation System Using Evolvable Hardware. *Artificial Life Journal,* 4(4), 359-393.

H. M¨uhlenbein and D. Schlierkamp-Voosen (1993). Predictive Models for the Breeder Genetic Algorithm I. *Evolutionary Computation.* 1(1):25-50.

R. Pfeifer and C. Scheier (1999). *Understanding Intelligence* MIT Press, Cambridge, MA.

I. Rechenberg (1973). *Evolutionsstrategie.* Frommann-Holzboog, Stuttgart. Also printed in (Rechenberg, 1994).

I. Rechenberg (1994). *Evolutionsstrategie.* Frommann-Holzboog, Stuttgart.

R. Salomon (1996). Increasing Adaptivity through Evolution Strategies. *From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior.* 411-420.

R. Salomon and L. Lichtensteiger (2000). Exploring different Coding Schemes for the Evolution of an Artifi cial Insect Eye. *Proceedings of The First IEEE Symposium on Combinations of Evolutionary Computation and Neural Networks.* 10-16.

D. Schlierkamp-Voosen and H. M¨uhlenbein (1994). Strategy Adaptation by Competing Subpopulations. *Parallel Problem Solving from Nature (PPSN III).* 199-208.

H.-P. Schwefel (1995). *Evolution and Optimum Seeking.* John Wiley and Sons, NY.

H.-P. Schwefel (1997). Evolutionary Computation — A Study on Collective Learning. *Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics, vol. 2.* 198-205.