Unconstrained Evolution and Hard Consequences

CSRP 397

Ad ian Thompson, Inman Ha vey and
Philip Husbands

Decembe 1995

To appear in: E. Sanchez and M. Tomassini, editors, *Toward Evolvable Hardware*, Springer-Verlag Lecture Notes in Computer Science, 1996. Presented at the workshop of the same name, EPFL, Lausanne, 2–3rd October 1995.

Under many circumstances robustness in the presence of noise or hardware faults is a crucial factor, which can increase the number of trials needed. Noise is not always a problem, indeed it may have advantageous evolutionary effects.

We discuss the constraints that can be relaxed and the hard consequences that must be recognised, initially at a theoretical level. Then a real example of evolved hardware will be presented in the light of these discussions. A simple asynchronous digital circuit directly takes echo pulses from a pair of left/right sonars, and drives the two motors of a real robot, so that it exhibits a wall-avoidance behaviour in the real world. The complete sensorimotor control system (no pre- or post-processing) consists of just 32 bits of RAM and a few flip-flops, and is even tolerant to single-stuck-at faults in the RAM. The remarkable efficiency of this circuit can be attributed to the facts that it was evolved as a physical piece of hardware in the real world, and that many of the constraints on its dynamics were under evolutionary control. The rationale behind this experiment applies to many other kinds of system, including Field-Programmable Gate Arrays (FPGA's) [2].

The paper proceeds thus: Sections 2–11 discuss various aspects of artificial evolution. Sections 12–14 cover issues of noise, the relationship between simulation and reality, and fault tolerance. Sections 15–17 discuss the theory of Intrinsic Hardware Evolution. Sections 18 and 19 give a case study of a physical piece of hardware, intrinsically evolved in the real world as a robot controller. A final section summarises the discussion.

2 Evolution not Design

Human beings find it difficult to design complex systems, and the main heuristic used to make design easier is "Divide and Conquer". The complex whole is decomposed into smaller semi-independent parts or modules, which can be tackled one at a time. For this to work the modules must have minimal interactions between them, to allow any one to be tackled independently of the others. However, there are many complex systems which either do not have any such decomposition, or do not obviously show how they should be carved up.

If, however, some objective fitness function can be deTd(can)2461000.33000.3(to)f,up.

their

5 S ecies Evolution

In artificial evolution problems of varying dimensionality one should expect to have a genetically converged population, in effect a *species*, at all times. This contrasts with optimisation problems, where convergence in a GA signals the end of the search process. Using the example given above, where a hardware design being evolved could potentially have any number of components up to 20, during any single generation one should expect all the members of the population to have the same or a very similar number of components, for instance 10. What counts as *similarity* will be qualified later.

The conceptual framework of SAGA was introduced by Harvey in 1991 in order to try to understand the dynamics of a GA when genotype lengths are allowed to increase [4]. It was shown, using concepts of epistasis and fitness landscapes drawn from theoretical biology [5], that progress through such a genotype space will only be feasible through relatively gradual increases in information in the genotype (typically, in genotype length). A general trend towards increase in length is associated with the evolution of a *species* rather than global search. Such evolutionary search in the space of hardware designs would be from initially simple designs for simple tasks, towards more complex designs for more complex tasks; although in natural evolution there is no externally provided sense of direction, in artificial evolution this can be provided.

Throughout such artificial evolution, a species will be relatively fit, in the sense that most members of the population will be fitter than most of their neighbours in the fitness landscape. Evolutionary search can be thought of as searching around the current focus of a species for neighbouring regions which are fitter (or in the case of neutral drift, not less fit) while being careful not to lose gains that were made in achieving the current *status quo*. In the absence of any mutation (or change-length) genetic operator, selection will concentrate the population at the current best. The smallest amount of mutation will hill-climb this current best to a local optimum. As mutation rates increase, the population will spread out around this local optimum, searching the neighbourhood; but if mutation rates become too high then the population will disperse completely, losing the hill-top, and the search will become random. If an ideal balance is achieved between selective forces and those of mutation (as modified by recombination), then some elements of the population can crawl down the hill far enough to reach a ridge of relatively high selective values. As discussed in [6], this results in a significant proportion of the population working their way along this ridge under selection, and making possible the reaching of outliers ever further in Hamming-distance in that particular direction from the current fittest. The term 'ridge' is used here to fit in with intuitive notions of fitness landscapes; in fact in high-dimensional search spaces such ridges may form complex neutral networks, percolating long distances through genotype space.

If any such outliers reach a second hill that climbs away from the ridge, then parts of the population can climb this hill. Depending on the difference in fitness and the spread of the population, it will either move *en masse* to the new hill as a better local optimum, or share itself across both of them.

So in a SAGA setup of evolution of a converged species, we want to encourage through the genetic operators such exploration along ridges to new hills, subject to the constraint that we do not want to lose track of the current hill. Eigen and co-workers use the concept of a *quasi-species* to refer to a similar genetically converged population in the study of early RNA evolution. To quote from [6]:

> In conventional natural selection theory, advantageous mutations drove the evolutionary process. The neutral theory introduced selectively neutral mutants, in addition to the advantageous ones, which contribute to evolution through random drift. The concept of quasi-species shows that much weight is attributed to those slightly deleterious mutants that are situated along high ridges in the value landscape. They guide populations toward the peaks of high selective values.

6 SAGA and Mutation Rates

Although progress of a species through a fitness landscape is not discussed in the standard GA literature, in theoretical biology there is relevant work in the related field of molecular quasi-species [7, 6]. In particular, analysis of the 'error catastrophe' shows that, subject to certain conditions, there is a maximum rate of mutation which allows a quasi-species of molecules to stay localised around its current optimum. Selection and mutation are opposing forces, the former tending to increase numbers of the fittest members of the population, while the latter tends to drag offspring down in fitness away from any local optimum. A zero mutation rate allows for no further local search beyond the current species, and other things being equal increased mutation rates will increase the rate of evolution. Hence if mutation rates can be adjusted, it would be a good idea to use a rate close to but less than any critical rate which causes the species to fall apart. A further possibility, in the spirit of simulated annealing, is to temporarily allow the rate to go *slightly above* the critical rate — to allow exploration — and then cut it back again to consolidate any gains thus made.

For an infinite asexual population, in the particular context of molecular evolution, Eigen and Schuster show [7] that these forces just balance for a mutation rate

$$m = \frac{\ln \sigma}{l}$$

where l is the genotype length and σ is the *superiority* parameter of the *master sequence* (the fittest member of the population) — the factor by which selection of this sequence exceeds the average selection of the rest of the local fitness landscape, and hence the rest of the population. The diagrams they show for the very sharp cutoff at the critical rate refer to a fitness landscape with a single 'needle' peak for the master sequence, taking all the rest of the population to be equally (un-)fit; where the hill slopes more gently from the master sequence, the cutoff is less abrupt. For typical values of σ between 2 and 20, the upper limit of mutation before a quasi-species 'loses its grip' on the current hill would be between $0.7/l$ and $3/l$. For finite population size, there is some reduction in this critical mutation rate (the 'error threshold') [8], but for genotypes of length order 100, and populations of size order 100, the error threshold will be extremely close to that for an infinite population.

Since it is the natural logarithm of σ which enters into the equation for m, variations in σ of an order of magnitude do not affect $ln(\sigma)$ (and hence the error threshold) as significantly as variations in genotype length. In conventional GAs, choice of mutation rates tends to be a low figure, typically 0.01 or 0.001 per bit as a background operator, decided upon without regard to the genotype length. The SAGA framework means that mutation rates of the order of 1 per genotype are required when using linear rank selection or tournament selection as discussed below, subject to some qualifications concerning recombination and 'junk DNA'. These qualifications tend to increase the recommended rate to somewhere in the range 1 to 5 mutations per genotype, the idiosyncratic nature of fitness landscapes for different problems making it difficult to be more specific.

When applying such mutation rates in a sd020Td(fitness)TjessWhenu.7(op)-999.95ssioicncrat09.

local optimum than one's intuition based on 3-D landscapes might lead one to think. In addition, the percolation of such paths through sequence space tends to mean that it does not matter too much where in sequence space a converged population starts; under many circumstances it is possible to reach all possible fit regions from most starting points.

The SAGA selection and mutation rates encourage just such exploration through neutral drift in sequence space.

8 Selection

Selective forces need to be maintained at the same level throughout an evolutionary run, so as to balance mutational forces and maintain a similar degree of genetic convergence throughout. Basing selection directly on absolute fitness values does not achieve this, and some system based on ranking of the population must be used. This implies that the fittest member of the population has the same expected number of offspring whether it is far better than the rest, or only slightly better. Truncation selection (reproducing only from a top slice of the population) is one way of achieving this, but generally is too severe in restricting exploration by the less fit mutants. Less severe methods are recommended such as linear ranking; for instance giving the top ranker twice the average expected number of offspring, and reducing this amount linearly as one goes down the ranks, towards zero.

One way to achieve an effect comparable to linear ranking in a steady state occurred independently in two different lineages within the population to be combined into one which has both: something not possible with asexual reproduction. On the other hand, it allows parents with a detrimental mutation to

This effect is related distantly to that of stochastic resonance. The arguments are similar to those for the 'Baldwin effect' [10]. Practical examples of this phenomenon can be seen in [11, 12]. The rôle of noise in smoothing the transition from simulations to reality is under investigation [12].

13 The Use of Simulation

Artificial evolution requires the evaluation of members of a population over the course of many generations. Virtually all the time and expense is in performing these evaluations rather than in the genetic operations; optimising these latter operations has little practical use. The number of evaluations may be reduced by setting up the GA appropriately. Each individual evaluation should also not be unnecessarily long.

Simulations are often seen as attractive, offering the possibility of performing evaluations in faster than real time. Such simulations would need to be validated by testing evolved architectures in realised form at regular intervals. However there are a number of possible problems, depending on what precisely is to be simulated. Consider in turn the cases where simulations are of just the hardware, of both hardware and environment, and of just the environment.

Simulation of the hardware alone might be attempted if appropriate reconfigurable hardware was not available — simulations are likely to run more slowly than the real hardware. This is 'extrinsic' evolution, as opposed to 'intrinsic' evolution to be discussed later. Often detailed simulations of physical properties of hardware are too computationally expensive to be practical; evolution can then only be effective with the real hardware.

When both the hardware and the environment are to be simulated, it is frequently the latter that is much more complex. Often such simulations are simply not practical. For instance, when evolving control systems for visually guided robots [11], the visual world of a robot takes so much computing power to adequately simulate that testing in the real world is simpler and faster.

Consider finally the use of real hardware in a simulated environment. It has been suggested by de Garis [13] that where a hardware device such as a robot controller must be evaluated within some environment, then an environment simulation could be implemented in electronics situated next to the evolving hardware control system on a VLSI chip. It is suggested that this will allow the speeding up of each evaluation by perhaps many orders of magnitude. There are two serious doubts to be raised about this proposal.

The first is scepticism about the complexity of the simulation. For many real tasks (and that of a robot controller in a human environment is one example) the complexity of those crucial aspects of the environment which must be adequately simulated makes it infeasible; e.g., when the environment includes other entities of a complexity and speed comparable to that of the hardware itself.

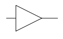
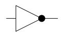
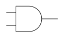
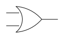
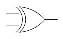
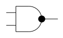
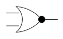
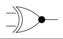
A second doubt arises when an adequate evaluator circuit running faster than real-time *can* be built (perhaps possible for simpler situations than robotics). The same hardware that has been evolved within a high-speed environment must then adjust to cope with normal timescales in the real world. This could be done with clocked hardware, by simply adjusting the clock-rate. However, the imposition of adjustable clocking on a piece of hardware is a severe constraint, limiting the range of dynamics available. Later sections will show the benefit of exhaustive search, on average. Thompson [14] deals with the evolution of fault tolerance in greater detail than here, also suggesting that the use of a co-evolving antagonistic population of emulated faults could force tolerance to a large set of faults in cases where the above technique is inapplicable or insufficient.

15 The Relationshi Between Intrinsic Hardware Evolution and Conventional Design Techni ues

Intrinsic[1] evolution of hardware means that the individuals of the evolving population receive fitness scores according to their performance when instantiated as real physical pieces of electronics.

15.2 Unconstrained Temporal Structure

Real physical electronic circuits are continuous-time dynamical systems. They can display a broad range of dynamical behaviour, of which discrete-time systems, logic nodes available was fixed at 100, and the genotype determined which of the boolean functions of Table 1(a) was instantiated by each node, and how the nodes were connected. The nodes were analogous to the reconfigurable logic blocks of an FPGA, but an input could be connected to the output of any node without restriction. The linear bit-string genotype consisted of 101 segments (numbered 0..100 from left to right), each of which directly coded for the function of a node and the sources of its inputs, as shown in Table 1(b). (Node 0 was a special 'ground' node, the output of which was always clamped at logic zero.) This encoding is based on that used in [23]. The source of each input was specified by counting forwards/backwards along the genotype (according to the 'Direction' bit) a certain number of segments (given by the 'Length' field), either starting from one end of the string, or starting from the current segment (dictated by the 'Addressing Mode' bit). When counting along the genotype, if one end was reached, then counting continued from the other.

Name	Symbol
BUFFER	
NOT	
AND	
OR	
XOR	
NAND	
NOR	
NOT-XOR	

(a)

Bits	Meaning
0-4	Junk
5-7	Node Function

The objective was for node number 100 to produce a square wave oscillation of 1kHz, which means alternately spending 0.5×10^{-3} seconds at logic **1** and at logic **0**. If k logic transitions were observed on the output of node 100 during the simulation, with the n^{th} transition occurring at time t_n seconds, then the average error in the time spent at each level was calculated as :

$$\text{average error} = \frac{1}{k-1} \sum_{n=2}^{k} \mid (t_n - t_{n-1}) - 0.5 \times 10^{-3} \mid \qquad (1)$$

For the purpose of this equation, transitions were also assumed to occur at the very beginning and end of the trial, which lasted for 10ms of simulated time. The fitness was simply the reciprocal of the average error. Networks that oscillated far too quickly or far too slowly (or not at all) had their evaluations aborted after less time than this, as soon as a good estimate of their fitness had been formed. The genetic algorithm used was a conventional generational one [3], with elitism and linear rank-based selection. At each breeding cycle, the 5 least fit of the 30 individuals were killed off, and the 25 remaining individuals were ranked according to fitness, the fittest receiving a fecundity rating of 20.0, and the least fit a fecundity of 1.0. The linear function of rank defined by these end points determined the fecundity of those in-between. The fittest individual was copied once without mutation into the next generation, which was then filled by selecting individuals with probability proportional to their fecundity, with single-point crossover probability 0.7 and mutation rate 6.0×10^{-4} per bit.[3]

Fig. 1 shows that the output of the best individual in the 40^{th} generation (Fig. 2) was approximately $4\frac{1}{2}$ thousand times slower than the best of the random initial population, and was six orders of magnitude slower than the propagation delays of the nodes. In fact, fitness was still rising at generation 40 when the experiment was stopped because of the excessive processor time needed to simulate this kind of network. This result suggests that it *is* possible for evolution to arrange for a network of high-speed components to generate much slower behaviour, without having to provide explicit 'slowing-down' resources with large time constants, and without the need for a clock (though these could still be useful).

The evolved oscillators produced spike trains rather than the desired square wave. Probing internal nodes indicated that this was because *beating* between spike trains of slighw(could)T9,

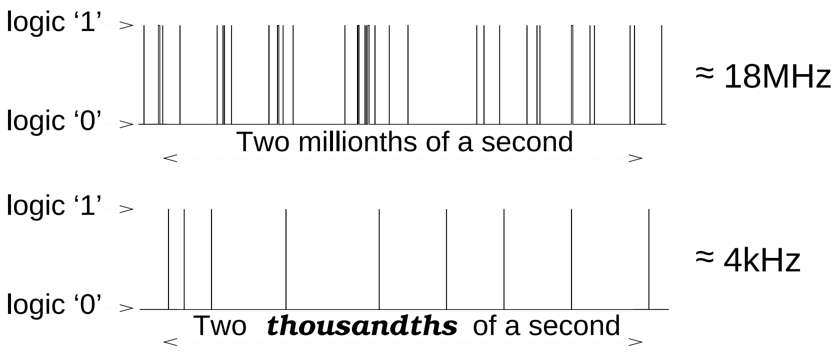


Fig. 1. Output of the evolving oscillator. (Top) Best of the initial random population of 30 individuals, (Bottom) best of generation 40. Note the different time axes. A visible line is

throughout the network, which does not have any significant modules[5].

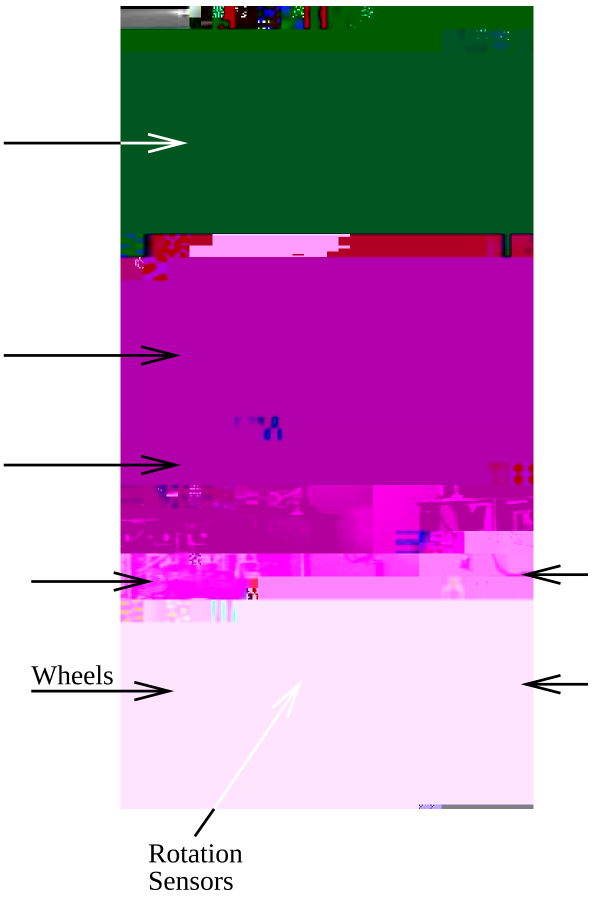
16 Ex loiting the Hardware versus Tolerance to Device Variations

In the evolved oscillator experiment of the previous section, each node had a slightly different input $\Rightarrow$ output time delay, crudely modelling the propagation delays of the reconfigurable blocks of an FPGA. The delays were initially randomly chosen but then held fixed during the experiment. If the delays of the final evolved circuit were *re-randomised* then the behaviour was totally destroyed, and the circuit was no better than a randomly generated one: it relied on the particular time delays present during its evolution. Extrapolating to real intrinsic hardware evolution, it can be expected that all of the detailed physics of the hardware will be brought to bear on the problem at hand: time delays, parasitic capacitances, cross-talk, meta-stability constants and other low-level characteristics might all be used in generating the evolved behaviour.

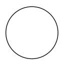
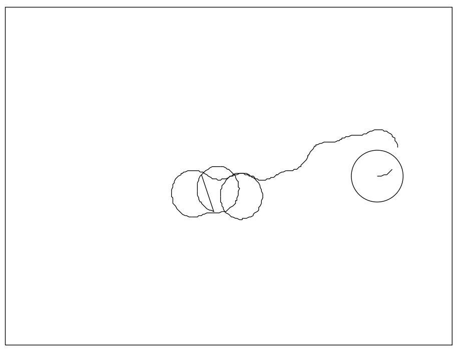
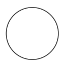
The exploitation of all of the hardware's physical properties must be traded against sensitivity to variations in them. Some tolerance is essential because of the inevitable changes over time due to fluctuations in the tem9.7(due)-11ecause cannot proceed far with precise descriptions of the physics of the individual components (eg. transistors) before abstraction and modularisation need to be invoked to make the problem tractable, as discussed above. The 'try it and see' opportunistic nature of intrinsic EHW is not subject to this difficulty, so the detailed behaviours of the components can be integrated usefully to give rise to the required overall performance. This paper largely concentrates on the use of digital reconfigurable devices because these are currently available off-the-shelf, but we can now see that the advantages of intrinsic EHW over conventional design are even greater for analogue systems. Analogue FPGAs are being developed [25] and will be a fruitful avenue for EHW research in the future.

18 Case Study: An evolved hardware sensorimotor control structure

This experiment takes a standard electronic architecture, removes some of the dynamical constraints used to make conventional design tractable, and sub


Wheels
Rotation
Sensors

[redacted] [redacted]

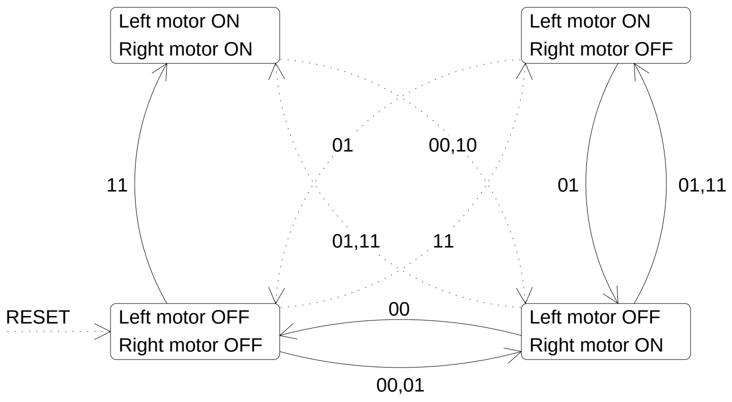


Fig. 7. A representation of one of the wall-avoiding DSMs. Asynchronous transitions are shown *dotted*, and synchronous transitions *solid.* The transitions are labelled with *(left, right)* sonar input combinations, and those causing no change of state are not shown. There is more to the behaviour than is seen immediately in this state-transition diagram, because it is not entirely a discrete-time system, and its dynamics are tightly coupled to those of the sonars and the rest of the environment.

probability of the machine being in a certain state at that same instant (remember that one of the feedback loops is unclocked).

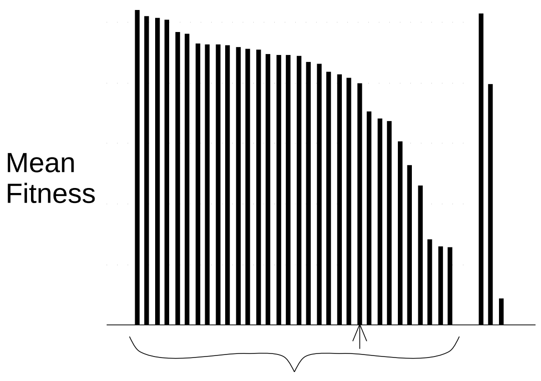
Even this small system is non-trivial, and performs a difficult task with minimal resources, by means of its rich dynamics and exploitation of the real hardware. After relaxing the temporal constraints necessary to support the designer's finite-state model, a tiny amount of hardware has been able to display rather surprising abilities[7]. FPGAs are undoubtedly very much more powerful than the DSM. It is left to the reader to speculate on the potential capabilities of an evolvable chip containing many hundreds of logic gates, bearing in mind the power released by unconstrained evolution from the equivalent of only two gates in the DSM described above.

19 The Fault Tolerance of The Evolved Hardware Control System

To end the paper, we briefly use the evolved DSM robot controller to consolidate two of the points made earlier pertaining to fault tolerance.

Firstly, notice that the contents of the RAM were directly encoded bit-for-bit onto the genotype. A genetic mutation in the RAM coding region causes one of

[7] If all of the genetic latches were constrained to be synchronous, so that we have a FSM, then three control experiments failed to produce successful systems. If all of the latches were constrained to be asynchronous, then three control experiments each produced a recognisable wall-avoider, but much inferior to those evolved when the genetic latches were under evolutionary control.


Mean
Fitness

20 Conclusion

We have formulated Evolvable Hardware — and in particular *intrinsic* EHW — as a synthesis of genetic algorithms, inspiration from natural evolution and electronics. In forming this synthesis, none of the three components has been left unaltered. For the evolution of complex structures, genetic algorithms need to be extended to incorporate the notions of converged species evolution and incrementally increasing complexity: the SAGA framework. When drawing inspiration from nature, it is necessary to recognise the significantly different constraints and opportunities associated with the artificial evolution of VLSI circuits. Finally, the whole concept of *what electronics can be* needs to be re-thought, because until now it has been governed by what is amenable to design techniques. Stepping into the wider space of dynamical electronic systems, the first intrinsically evolved hardware robot controller has been presented, showing remarkable levels of efficiency and robustness. There is every reason to expect that this new field will go far, but extravagant projections are not appropriate at this early stage.

Acknowledgements

dradradrashoshoshoshoshoshoshovts

11. I. Harvey, P. Husbands, and D. T. Cliff. Seeing the light: Artificial evolution, real vision. In D. Cliff, P. Husbands, J.-A. Meyer, and S. Wilson, editors, *From Animals to Animats 3: Proc. of 3rd Int. Conf. on Simulation of Adaptive Behaviour (SAB94)*, pp392–401. MIT Press/Bradford Books, Cambridge MA, 1994.
12. N. Jakobi, P. Husbands, and I. Harvey. Noise and the reality gap: The use of simulation in evolutionary robotics. In F. Morán et al., editors, *Advances in Artificial Life: Proc. of 3rd Eur. Conf. on Artificial Life (ECAL95).*, pp704-720. LNAI 929, Springer-Verlag, 1995.
13. Hugo de Garis. Evolvable hardware: Genetic programming of a Darwin Machine.