

Notes on Design Through Artificial Evolution: Opportunities and Algorithms

Adrian Thompson¹

Evolutionary & Adaptive Systems Group,
University of Sussex,
Brighton BN1 9QH, UK
adrianth@cogs.susx.ac.uk

Abstract. An attempt is made to isolate a class of design problems that only evolutionary methods can tackle. A preliminary evolved design for a nano-electronic circuit is found to contain a switching element that relies on the stochastic passage of electrons due to thermal noise. Such phenomena have been exploited by natural evolution in neural systems, but not before for circuit design. There is room for an imaginative leap into areas of design space only accessible through evolution. Analysis of the evolution of a second circuit reveals that neutral evolution played a key role, and can be a natural property of evolutionary design. The developing theory of evolutionary design promises practical future benefits.

1 Introduction

There are times when artificial evolution can be applied profitably even though more traditional methods are available. It is also possible for evolution to produce designs that could not practically be arrived at any other way. In the latter case, creativity is needed to experiment with potential applications, because the possibility of such designs has not existed before. In tandem, careful analysis of what exactly evolution *can* do is required. These notes aim to give some insights into evolution beyond the scope of conventional design. Examples (intended to be intriguing) are drawn from the author's work on electronics design, but they illustrate principles for design in general.

The notes are divided into two parts. 'Opportunities' offers some remarks on the evolution of radically unconventional designs, and gives an example of an evolved nano-electronic design that employs dynamical principles previously seen in the literature of neuroscience, but not of nano-electronics. The 'Algorithms' section then provides food for thought on what types of evolutionary algorithm may be able to tackle challenging design problems in the future, given developments in computer architecture. An example shows how a very simple mutation-driven algorithm can arrive at a surprisingly sophisticated design if neutral evolution is allowed.

2 Evolutionary Design: Opportunities

For the sake of discussion, let's agree that evolution consists of selection acting repeatedly on heritable variation, where that variation is essentially blind, rather than incorporating detailed heuristics.

For design problems, the evolutionary algorithm (EA) determines some of the structure and/or parameters of a reconfigurable object. This object may exist in software, though that could be a simulation of the hardware of an eventual implementation. The reconfigurable object might alternatively be physically changeable hardware.

Typically, the object is embedded in some sort of environment, to which it responds, which it influences, or in which it behaves. The EA designer devises a fitness evaluation procedure that monitors and possibly manipulates the environment and object, and returns objective function values.

In the examples to follow, the object is an electronic circuit, either simulated, or as a field-programmable gate array (FPGA) — a physically reconfigurable integrated circuit. The circuit behaves in the environment when supplied with test inputs, and the evaluation procedure induces those inputs and measures the quality of the behaviour. The evaluation may also include non-behavioural measurements of the object itself, such as size and power consumption.

A more subtle case would be the evolution of a musical score that is to provoke a human to report a particular kind of feeling [1]. The musical score is the reconfigurable object, but it is the behaviour of the whole immensely more complex system that is crafted by evolution.

Figure 1 shows how the situation appears to the evolutionary algorithm. It generates structural/parametric variations of the object, by applying variation operators (such as mutation and crossover) to some representation of the object's configuration. All it gets back are the measured objective values: we can think of the entire evaluation/environment/object complex as a black-box system.

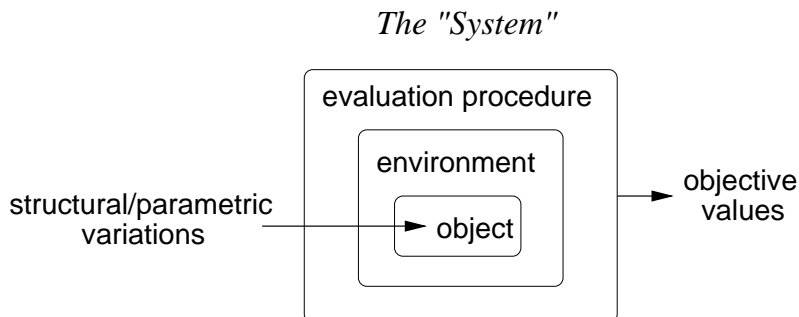


Fig. 1. A view of an evolutionary algorithm coupled to a 'black box' system.

Defining this black-box system allows us to consider three separate cases that tease apart the differences between evolutionary and conventional design:

- a) **Inverse model is tractable:** If there is a tractable ‘inverse model’ of the system, then there is a way of working out in advance a sequence of variations that brings about a desired set of objective values. ‘Conventional’ methods can be applied: the blind generate-and-test nature of evolution is not essential, though it still may be that evolutionary methods are competitive.
- b) **Inverse model is not tractable, but forward model is:** In this case, we can predict the influence of variations upon the objective values, but the system is not tractably invertible so we cannot derive in advance a sequence of variations to bring about a desired set of objective values. This implies an iterative approach, where variations carefully selected according to the forward model are applied in sequence. This kind of iterative design-and-test is a common component of traditional approaches. Search techniques, including evolutionary algorithms, can be competitive or in some cases the only viable choice [2].
- c) **Neither forward nor inverse models are tractable:** There is neither a way of discerning which variations will give improvements in the objective values, nor a way of predicting what will be the effects of variations upon the objective values. Without evolution all is lost. By our tentatively agreed definition, evolutionary methods are those that proceed by incrementally applying variations that are essentially *blind*. Selection can lead to an improvement in objective values with neither a forward nor an inverse model. Whether evolutionary methods actually succeed in finding a satisfactory design is another matter, but they are the only way to do it. Note that our definition of evolution encompasses methods that historically have not been related, such as simulated annealing.

Thus there is an entire class of design problems that can only be tackled by evolutionary methods. Practically speaking, where are these problems? There are few examples, because evolutionary design is in its infancy: previously such design problems have had to be avoided or circumvented. Figure 2 shows one example prior to the birth of the field of evolutionary computation.

We are now in a situation demanding a leap of imagination for EA practitioners. Previously, design problems had to be constrained or simplified to form type (a) or (b) systems. We can now explore a wider space of designs, though these will inevitably seem strange.

Consider the design for an electronic circuit shown in Figure 3. As circuit sizes are reduced towards the nano-scale, it becomes necessary to exploit quantum effects directly for computation and data storage, rather than attempting to suppress them in macroscopic approximations. How can quantum effects best be employed? The literature is full of ideas [5], but there is no consensus. This experiment, fully reported in [6], was a preliminary attempt

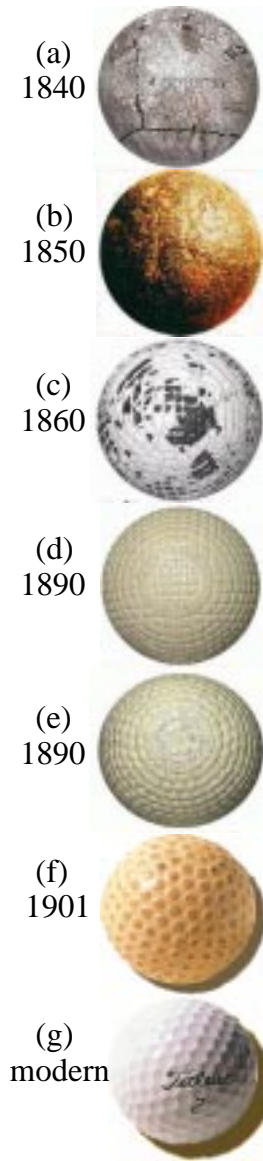


Fig. 2. Golf balls: one of the few examples of true evolutionary design outside of the field of evolutionary computation. Dates are approximate.

(a) is a ‘featherie’ made by packing an enormous quantity of feathers into a hide casing. These balls were expensive.

(b) is an early ‘guttie’ made of solid gutta percha (a rubberish tree sap). The primary advantage was lower cost, but these smooth balls were found not to travel as far as featheries.

It had been noticed that used featheries, which had acquired nicks and cuts, travelled further than new featheries. Adding surface texture to gutties was found to have a similar effect, and various patterns of cuts (c) and raised mouldings (d,e) were blindly but incrementally experimented with. The first balls with a wound rubber core (f) followed suit. Modern balls favour dimples (g), and it is only relatively recently that the aerodynamics causing a rough ball to travel further have been understood [3].

Interestingly, some of the very first explicit forays into artificial evolution also designed the aerodynamics of a physical system [4].

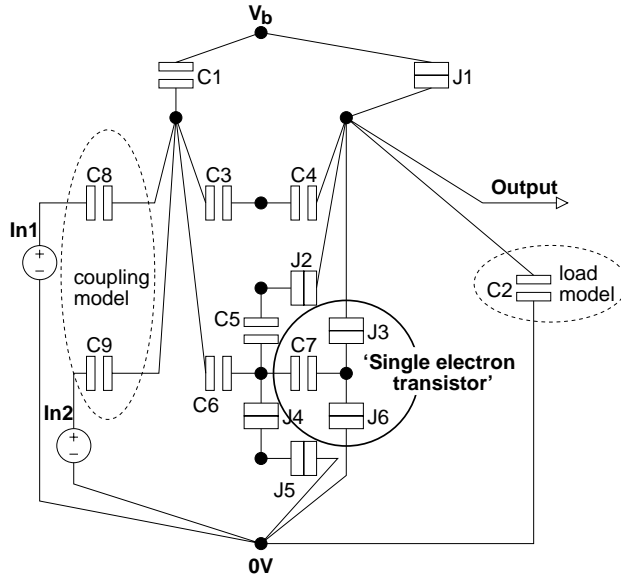


Fig. 3. An evolved ‘single electron’ NOR gate for a temperature of 340mK in a simplified scenario. Fixed values: C2 1.000e-12F; C8 3.333e-13F; C9 3.333e-13F. Evolved Values: C1 4.858e-19F; C3 9.969e-14F; C4 2.052e-16F; C5 1.000e-13F; C6 3.393e-16F; C7 2.975e-15F; J1 4.000e-19F 4.950e+06Ω; J2 4.000e-19F 5.766e+05Ω; J3 4.059e-19F 9.024e+04Ω; J4 4.237e-19F 5.854e+04Ω; J5 3.632e-16F 2.886e+07Ω; J6 4.857e-19F 5.000e+04Ω; V_b -1.000e-04V; V_{false} -8.368e-05V; V_{true} -8.488e-06V

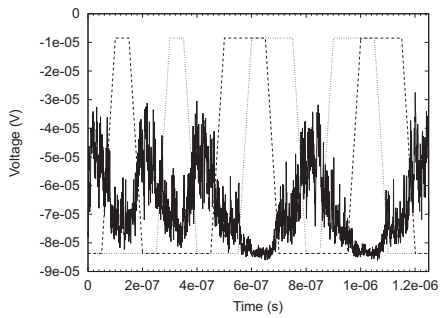


Fig. 4. The input/output relationship of the NOR gate. The dashed and dotted lines are the inputs, the solid line is the output.

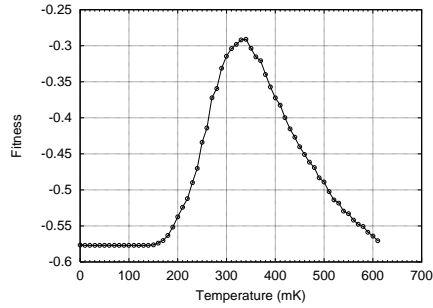


Fig. 5. The NOR gate’s performance as the temperature is varied.

to allow evolution to explore the design space as a type (c) system, with the minimum or simplifying constraints or prejudice.

The dots represent isolated conductive islands between which electrons can move through quantum-mechanical tunnelling. The size, shape, and position of the islands determines their capacitive coupling (marked C_n) and the resistance which characterises a tunnel junction (J_n). Although unrealistic in many respects, the experiment did respect the necessity of a local topology that allows interacting islands to be physically adjacent.

This simple NOR gate was evolved in simulation for a temperature of 340mK. This is cold, but many studies concentrate on analysis at 0K; even at 340mK the electrons have sufficient thermal energy (‘noise’) to challenge a designer. It is expected that the performance of a circuit will fall with rising temperature, but Figure 5 reveals that the evolved circuit’s behaviour also degrades as the temperature is *decreased* from 340mK. This kind of behaviour had never been seen in such proposed ‘single electron’ circuits before, and indicates that the circuit actually exploits or relies upon the thermal noise of the electrons at 340mK. This is not necessarily desirable, and perhaps by evaluating across a range of temperatures during evolution a thermally robust solution could be found [7], but we see immediately that evolution is exploring a previously inaccessible part of design space.

Circled in Figure 3 is the well-known configuration of a ‘single electron transistor’, which has subtle dynamics that act as a switch if used carefully. It does indeed act as the main switching element in this circuit, but at 0K no electrons ever have sufficient energy to pass through the switch, no matter what the circuit’s inputs are. The circuit relies on the thermal energy of the electrons at 340mK to excite them to pass through the switch in a stochastic manner, influenced by the inputs. This type of phenomenon is found in a variety of physical systems [8], and has been exploited by natural evolution in some neural systems, but not before in electronic circuit design.

There is room for debate in the discussion above, but the argument is compelling that we are now faced with an imaginative leap into new territories of design space that are accessible through evolution. Some other explorations in evolutionary electronics can be found in [9,10].

3 Evolutionary Design: Algorithms

A formal analysis of the computational complexity of an algorithm is not a perfect guide to real-world performance. For example, the complexity of sorting algorithms has been much studied, but the order in which they make memory accesses can be central to performance on a modern computer having a particular architecture of memory caches [11]. Similarly, it may be that the evolutionary algorithms of choice for large-scale design problems in the future will be those that can exploit parallel architectures, such as ‘Beowulf’ networks of cheap off-the-shelf PCs connected by low-bandwidth links [12].

What kinds of evolutionary algorithms are capable of producing a complex design? An experiment fully reported in [7] evolved the configuration of an FPGA chip for a simple tone-discrimination task. The task was challenging because no prior rules were enforced regarding how the logic cells on the chip could be connected, but the fitness evaluations required that configurations should perform well on a variety of samples of the chip at different temperatures. Without the constraints of digital design rules, most configurations of the chip result in badly-behaved circuits, perhaps displaying wild oscillations or an extreme sensitivity to the characteristics of the silicon or its temperature. The experiment was to determine whether a robust solution could be found that did not simply rediscover conventional design rules.

For a fitness evaluation, a variety of chips needed to be configured and monitored: for example, one was heated, and another was in a freezer. The chips ought to be well electrically isolated from each other to prevent mutual interference. These factors meant that the best way of configuring the chips was over a serial interface needing few wires, but that is slow. It was decided to try a simple (1+1) evolution strategy (ES), so that only single mutations needed to be sent to the chips most of the time, rather than full configurations. The ES maintained a single individual, and tested mutants of it. If a mutation caused a reduction in fitness, it was rejected, otherwise it was accepted. This is simply a random-ascent hill climber allowing neutral moves, but has the essential ingredients of evolution.

The experiment succeeded, and a robust asynchronous design was found that could not have resulted from normal design principles. Why did the simple mutation-driven ES not get stuck on a local optimum? Consider Figure 6. The circuits have been empirically pruned to show only the components and connections involved in generating their behaviour. Circuit (a) is near the beginning of a fitness plateau over which many neutral mutations are made. (Some evaluation noise is visible, but the underlying fitnesses on the plateau were equivalent.) It turns out that if we test all possible single mutations of circuit (a), none of them result in improved fitness. At first glance, this might suggest that the simple ES is now stuck. Thousands of neutral mutations are then made, resulting in circuit (b) of equal fitness. Notice that these mutations did not merely affect ‘junk’ circuitry around the periphery, but made significant changes to the structure of the functional core shown here. Then circuit (b) acquired a single mutation to give circuit (c), now with a higher fitness. That single mutation was the connection shown in bold running across the lowest FPGA cell in (c). The connections shown dotted in (b) are not involved in generating behaviour, but in (c) they are. The neutral drift has moved us from a circuit that cannot be improved upon by single mutations to one that can, and eventually an improving mutation was found.

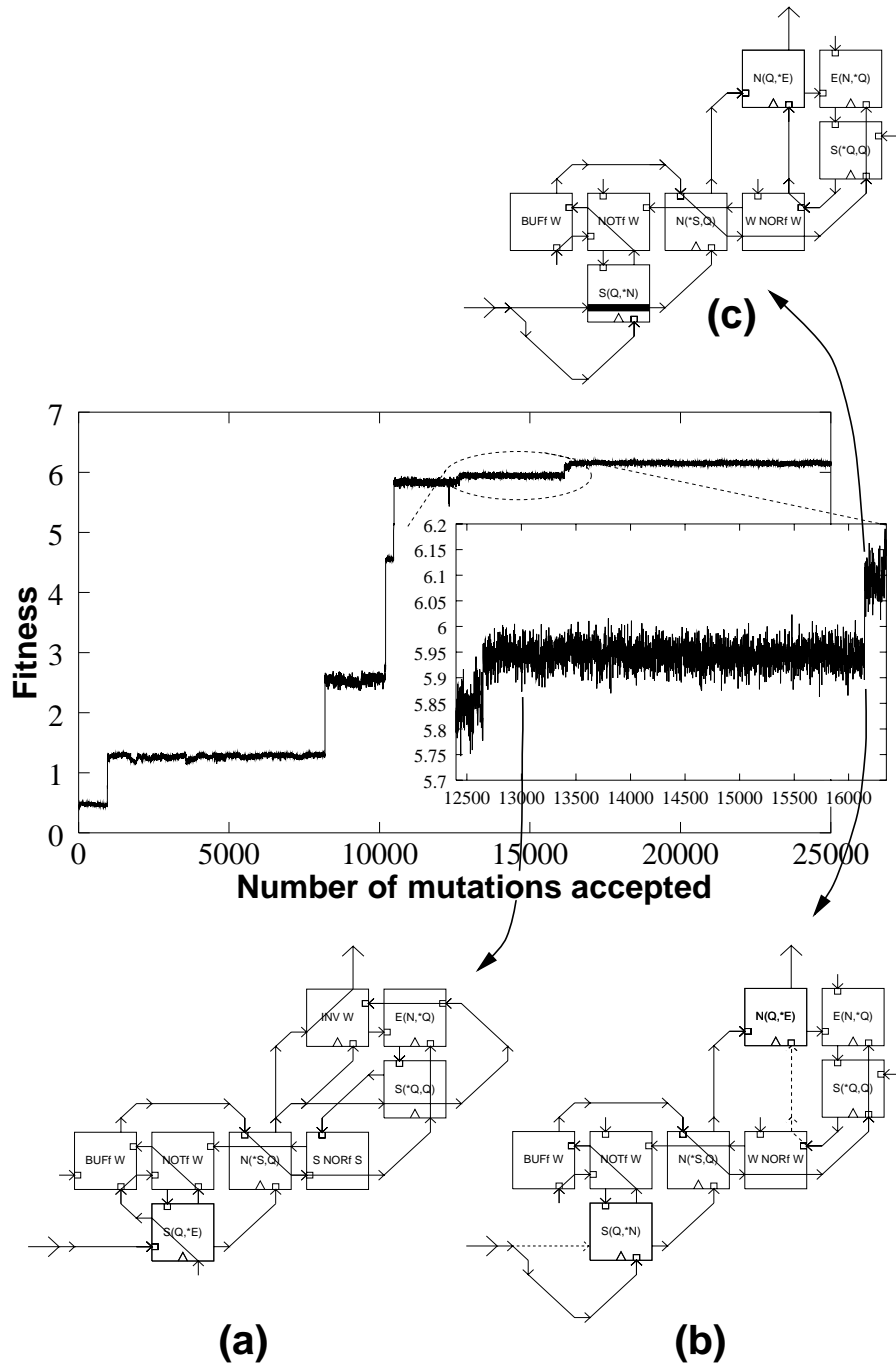


Fig. 6. (*Centre*) The fitness graph for a (1+1)ES used to design the configuration of a field-programmable gate array chip. (*a*) The functional part of the circuit at 13000 mutations, near the beginning of a fitness plateau. (*b*) The functional part of the circuit at the end of the plateau, 3144 mutations later. (*c*) The functional part of the circuit 1 mutation later, now with a higher fitness.

No claim is made that this evolutionary run was efficient, but given its desirability for hardware convenience it was able to work. The analysis shows that neutral variations played a causal role in the course of evolution. The ability to make neutral variations was a natural outcome of the application of evolutionary variation operators to a design object, and is not peculiar to this experiment. Neutrality is only recently coming to be discussed in the evolutionary computation community (e.g. [13,14]), but appears to be an inherent property of evolutionary design. Despite the apparent maturity of the field of evolutionary computation, it is heartening that further theoretical and empirical analysis of evolutionary design can be of practical power.

4 Conclusion

Given that the field of evolutionary computation is now fairly mature, it is tempting to think that there is a good understanding of what evolutionary design can achieve, and how. These notes have aimed to suggest the contrary.

An argument was offered that there is a whole class of design problems that only methods with an evolutionary flavour can tackle. Designs of this kind are found in nature, but not before for artifacts except in rare cases. An example of an evolved nano-electronic circuit showed a dynamical effect generally only exploited in the works of natural evolution. It was then seen that neutral variation can be a natural part of evolutionary design that can crucially affect the outcome. Without making any grand claims for the potential achievements of artificial evolution in the future, it seems that the enterprise is still young.

Thompson is grateful for an EPSRC Advanced Research Fellowship. Special thanks to Phil Husbands, Inman Harvey, Paul Layzell, and Christoph Wasshuber.

References

1. G. L. Nelson. Sonomorphs: An application of genetic algorithms to the growth and development of musical organisms. In *Proc. 4th Biennial Art & Technology Symp.*, pages 155–169, Connecticut College, March 4–7th, 1993. <http://timara.con.oberlin.edu/~gnelson/papers/morph93/morph93.htm>.
2. P. P. B. de Oliveira, F. M. Ramos, R. C. Gatto, et al. A research agenda for iterative approaches to inverse problems using evolutionary computation. In *Proc. 3rd IEEE Int. Conf. on Evolutionary Computation*, pages 55–60. IEEE Press, Piscataway NJ, 1996.
3. M. Fischetti. Flight control. *Scientific American*, June 2001.
4. I. Rechenberg. Cybernetic solution path of an experimental problem. Royal Aircraft Establishment, Library Translation 1122, 1965. Reprinted in ‘Evolutionary Computation — The fossil record’, D. B. Fogel, ed., chap. 8, pp297-309, IEEE Press 1998.

5. I. Csurgay, W. Porod, and S. M. Goodnick, editors. *Int. J. Circ. Theor. Appl.* Special issues on nanoelectronic circuits. John Wiley & Sons, 2000/1. Vol. 28 Issue 6 & Vol. 29 Issue 1.
6. A. Thompson and C. Wasshuber. Design of single-electron systems through artificial evolution. *Int. J. Circ. Theor. Appl.*, 28(6):585–599, 2000.
7. A. Thompson and P. Layzell. Evolution of robustness in an electronics design. In J. Miller, A. Thompson, P. Thomson, and T. Fogarty, editors, *Proc. 3rd Int. Conf. on Evolvable Systems (ICES2000): From biology to hardware*, volume 1801 of *LNCS*, pages 218–228. Springer-Verlag, 2000.
8. L. Gammitoni, P. Hänggi, P. Jung, and F. Marchesoni. Stochastic resonance. *Reviews of Modern Physics*, 70(1):223–287, 1998.
9. J. Miller, A. Thompson, P. Thomson, and T. Fogarty, editors. *Proc. 3rd Int. Conf. on Evolvable Systems (ICES2000): From Biology to Hardware*, volume 1801 of *LNCS*. Springer-Verlag, 2000.
10. A. Thompson, P. Layzell, and R. S. Zebulum. Explorations in design space: Unconventional electronics design through artificial evolution. *IEEE Trans. Evol. Comp.*, 3(3):167–196, 1999.
11. A. LaMarca and R. E. Ladner. The influence of caches on the performance of sorting. *Journal of Algorithms*, 31(1):66–104, 1999.
12. F. H. Bennett III, J. R. Koza, J. Shipman, and O. Stiffelman. Building a parallel computer system for \$18,000 that performs a half-petaflop per day. In W. Banzhaf, J. Daida, A. E. Eiben, et al., editors, *Proc. Genetic and Evolutionary Computation conference (GECCO-99)*, pages 1484–1490. Morgan Kaufmann, 1999.
13. I. Harvey and A. Thompson. Through the labyrinth evolution finds a way: A silicon ridge. In T. Higuchi, M. Iwata, and L. Weixin, editors, *Proc. 1st Int. Conf. on Evolvable Systems (ICES'96)*, volume 1259 of *LNCS*, pages 406–422. Springer-Verlag, 1997.
14. L. Barnett. Netcrawling — optimal evolutionary search with neutral networks. In *Proc. Congress on Evolutionary Computation (CEC)*, pages 30–37. IEEE, 2001.