Through the Labyrinth Evolution Finds a Way: A Silicon Ridge

Inman Harvey and Adrian Thompson*

School of Cognitive and Computing Sciences,
University of Sussex,
Brighton, BN1 9QH, UK

Abstract. Artificial evolution is discussed in the context of a successful experiment evolving a hardware configuration for a silicon chip (a Field Programmable Gate Array); the real chip was used to evaluate individual configurations on a tone-recognition task. The evolutionary pathway is analysed; it is shown that the population is genetically highly converged and travels far through genotype space. Species Adaptation Genetic Algorithms (SAGA) are appropriate for this type of evolution, and it is shown how an appropriate mutation rate was chosen. The role of junk on the genotype is discussed, and it is suggested that neutral networks (paths through genotype space via mutations which leave fitness unchanged) may be crucial to the effectiveness of evolution.

1 Introduction

To evolve silicon hardware successfully one must understand the limitations and possibilities of hardware; but one must also understand the limitations and possibilities of artificial evolution. This paper focuses on these latter concerns, and illustrates them with data taken from the first ever successful experiment evolving circuit designs directly onto a Field Programmable Gate Array for the purpose of signal recognition, using intrinsic evolution.

Newcomers to the field of hardware evolution tend to draw their ideas about artificial evolution from the Evolutionary Algorithm (EA) literature. Over the last 20 years a body of practice has accumulated, applying algorithms based upon evolutionary methods to a wide range of optimisation problems. It will be suggested here that hardware evolution will typically require a subtly but very significantly different methodology from that portrayed in much of the EA literature. We claim that the usual fears of premature convergence of the population are groundless; that entrapment at local optima does not usually take the form that is conventionally pictured; and neutral networks through genotype space should be encouraged through the artificial equivalent of potentially useful junk DNA.

* Emails: inmanh, adrianth @cogs.susx.ac.uk
WWW: http://www.cogs.susx.ac.uk/users/inmanh, adrianth

Section 2 of this paper will summarise the hardware experiment, the pattern of fitnesses exhibited during the run, and the final evolved hardware configuration. The degree of genetic convergence in the population is plotted, showing that a high degree of convergence is achieved almost immediately and then maintained. This runs counter to conventional EA expectations, where pains are often taken to prevent this on the (usually false) assumption that such convergence implies there will be no further evolution. In Section 3 the background assumptions of mainstream EAs are examined; Section 4 suggests why this often leads to false assumptions about genetic convergence.

In Section 5 the concepts of neutrality and neutral networks within a fitness landscape are introduced. Section 6 discusses the role and usefulness of junk or redundancy in genotypes, and suggests that some forms of junk should be encouraged in hardware evolution.

Having given a theoretical context, we then return to the data from the real experiment in Section 7, where the genetic variation is plotted. The movement of the population through genotype space is plotted in Section 8, and we discuss the speed of this movement during evolution. We give a baseline comparison with a similar population not under directed selection. There are indications that when under selection, the population 'searches' along high-fitness ridges or neutral networks within the fitness landscape. In Section 9 we are able to give a sketch of the fitness landscape, and relate it to the evolutionary pathway.

Before drawing general conclusions, Section 10 discusses the philosophy behind Species Adaptation Genetic Algorithms (SAGA), where GAs are adapted to the sort of conditions that will usually hold in hardware evolution: the continued evolution, without entrapment, of a genetically converged population or 'species'.

2 The Hardware Evolution experiment

In a companion paper [26] Thompson describes in detail the intrinsic hardware evolution directly onto a Field Programmable Gate Array (FPGA) of a circuit which is required to distinguish between two input tones of 1kHz and 10kHz. The natural timescale of signals within the FPGA, used without any clocking, is of the order of nanoseconds, hence this task is difficult with the limited resources made available. Here the experiment is summarised in sufficient detail to give a context for the analysis of the dynamics of the evolutionary process.

A genotype of 1800 bits directly encoded the functions and pattern of connections of 100 logic blocks, or 'cells', within the FPGA: 18 bits for each cell. A population of size 50 was initialised randomly. In a generational genetic algorithm (GA) each genotype was used to reconfigure the FPGA which was then evaluated at the task. The next generation was generated by first copying over the elite member unchanged; the remaining 49 members were derived from parents chosen through linear rank-based selection, in which the fittest individual of the current generation had an expectation of twice as many offspring as the median-ranked individual. Single-point crossover probability was 0.7, and the

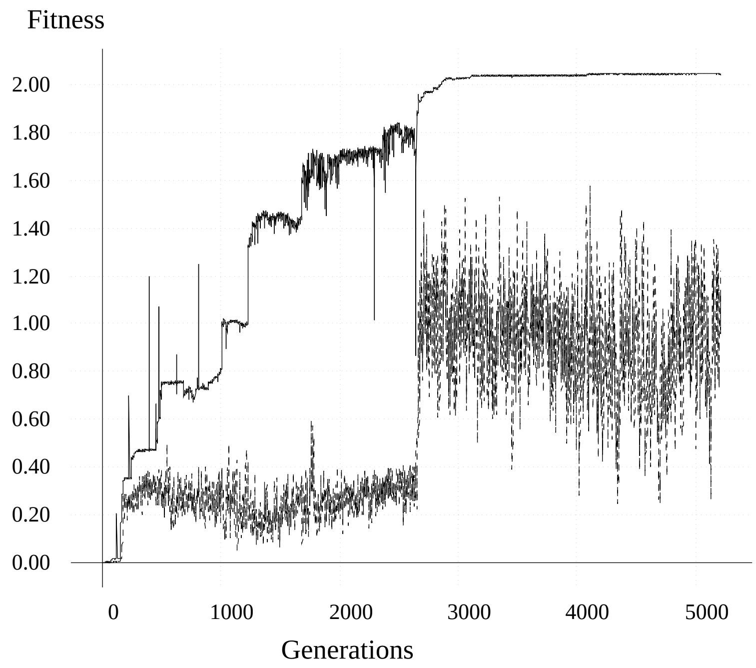

Fig. 1. The maximum and mean fitnesses of the population are plotted over 5220 generations. The dramatic rise in mean fitness occurs at generation 2660.

per-bit mutation rate was set such that the expected number of mutations per genotype was 2.7.

Evolution was continued for 5220 generations, with full genetic data saved every 10 generations. In Figure 1 the maximum and mean fitnesses are plotted, showing a dramatic increase in the mean at around 2660 generations, and the maximum fitness reaching a plateau at around 3000 generations (there is a small but definite further improvement shortly after generation 4000). This fitness corresponds to near-perfect performance at discriminating between the two input tones, signalling the result by a high or low output signal.

The fittest hardware design of generation 5000 is shown in Figure 2, where the arrows show the genetically specified pattern of connections between each of the 100 available cells (though not the functions performed by each cell). Systematic testing demonstrates that it is only a group of cells in the top left corner, between the input and output nodes, that is functional in the performance of the task. All of the other cells can be simultaneously clamped to fixed values without affecting the behaviour. The relevant functional cells are shown on the right in Figure 2. Some 2/3 of the cells, left blank in the right-hand figure, are irrelevant in the context of the functional part; hence mutations altering such irrelevant connections will be neutral with respect to fitness, except possibly for some immediately around the periphery of the functional part.

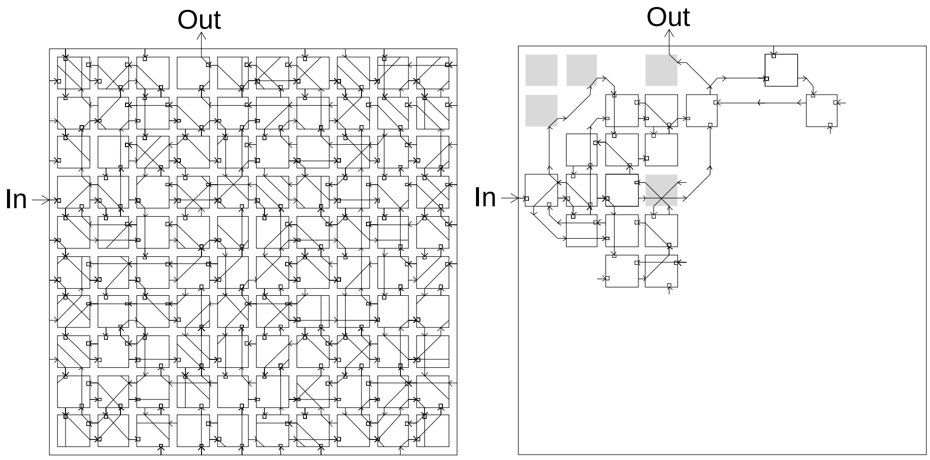

Fig. 2. On the left is shown the genetically specified configuration of the FPGA after evolution for 5000 generations. On the right is shown the subset of these connections which is functional, in the sense that the hardware still functions appropriately when the rest of the chip is clamped to fixed values.

This implies that for this particular functional design, up to 2/3 of the 1800 bits could be mutated without affecting the fitness. One can also realistically expect that there may be many other unrelated functional designs with a comparable number of redundant bits. Thus it would be a conservative estimate to suggest that out of the 2^{1800} possible points in genotype space — binary genotypes of length 1800 — at least 2^{1200} points represent hardware designs successful at the task.

In Figure 3 we plot on the left side the genetic convergence within the population as evolution progresses. This is measured as the average Hamming distance between pairs of genotypes drawn from the population. In the initial random population of genotypes of length 1800, this average is around 50% or 900 bits, but it can be seen that genetic convergence to below the 100 level is rapid, occurring within the first 45 generations. There is a temporary climb to above the 200 level after 2000 generations, which then falls back at around 2660 generations: the same time as the sudden rise in mean fitness shown in Figure 1.

The rapid convergence is not surprising, because something similar happens even in the absence of selective forces, merely through random genetic drift. In [1] it is shown that with a population of size N, with zero mutation, uniform recombination of n binary loci, and random selection of parents the mean convergence time to zero variation is approximately $1.4N(0.5log_e\mathrm{n}+1.0)^{1.1}$ generations. In the presence of mutation the convergence is not to a zero level of variation, but to a higher balance between selection/drift and mutation; this is reached considerably sooner. To give a baseline comparison to the hardware

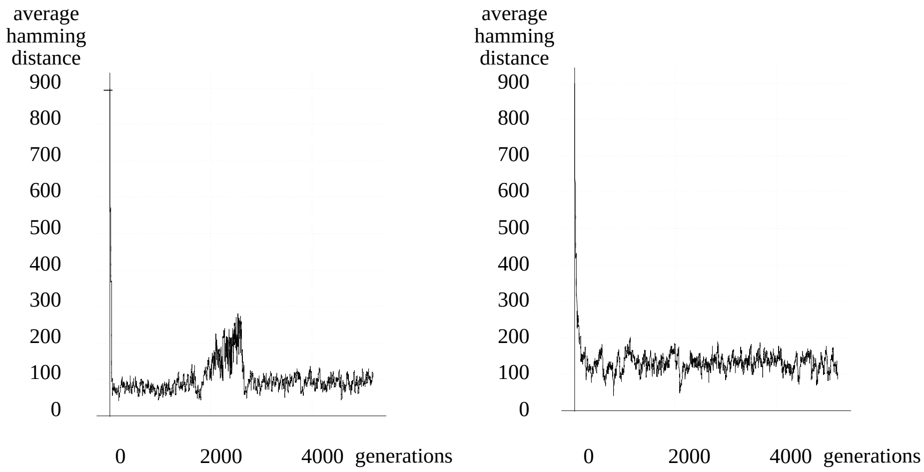

Fig. 3. Plot of genetic convergence within the population (measured as average Hamming distance between pairs of genotypes) against generations. On the left, from the evolving hardware; on the right, with fitnesses randomly allocated.

evolution example, an evolutionary experiment was run with the identical GA conditions, save only that the fitness of each genotype was allocated randomly at each test instead of being based on performance at the tone-discrimination task. The convergence statistics are shown on the right of Figure 3. In this case it takes some 220 generations to drop down to the 150 level; the average value is then maintained somewhat above the 100 mark.

Almost all of the improvement in fitness occurs after genetic convergence; the same phenomenon was discussed in the context of a different set of evolutionary experiments in [11]. For many users of GAs this is unexpected; the following sections will attempt to explain their surprise.

3 A Broad Picture of Evolutionary Algorithms

Historically there have been at least four main strands or flavours of evolutionary algorithms (EAs). The most prominent flavour is Genetic Algorithms (GAs) founded in the 1960s by John Holland [14, 9] where genotypes typically are strings of characters, often binary. Since the 1980s a prominent offshoot of GAs has been Genetic Programming (GP) [2, 21] where genotypes are typically simplified computer programs in the form of tree structures. Independent flavours include Evolution Strategies (ES) [23, 24] where the genotypes include real numbers; and Evolutionary Programming (EP) [7, 6]. Many concerns are shared across these paradigms; in this paper we shall focus on GAs, but the conclusions have wider applicability.

A common assumption is made within the GA community, explicitly or implicitly, that there are two distinct phases within an evolutionary run. The second and final stage is referred to as 'convergence', and the first stage is the transition from the (often randomised) starting population to the converged state.

There is a potential ambiguity in the word 'converged' which is regrettably often unrecognised. One meaning is that of genetic convergence: the state of the population where every genotype is similar to every other one. A second meaning is that the genetic search has converged onto its final resting place, perhaps a local optimum; often a fitness graph is shown with the fitness increasing to some asymptotic value, reaching it at 'the point of convergence'. It is frequently assumed by members of the GA and GP community that the two different types of convergence occur at one and the same time; hence the ambiguity is often considered to be of little importance.

There are some particular constrained circumstances in which this does indeed happen, but in general it does not. The evolutionary pathway of hardware design studied in depth in this paper is a clear case where the time of genetic convergence does not correspond to the time of convergence to a fitness optimum. In the general evolutionary case a population has a high degree of genetic convergence — it is a 'species' — with the degree of convergence maintained by a balance between selective pressures and genetic operators. Selective pressures act to increase convergence; genetic operators such as mutation act to increase diversity.

If there is any initial imbalance in favour of selection between these competing pressures, then a transitory period of genetic convergence occurs, often very rapidly. This does not in general mean that further evolution ceases.

4 Why is there this confusion about convergence?

Many GA implementations do fit into the particular constrained circumstances in which genetic convergence does indeed (exceptionally) imply no further evolutionary progress. When using a GA to optimise a function of n variables, a common strategy would be to use m bits to code for each variable on a genotype of length nm; the encoding of each variable to m-bit precision could be in binary or using a Gray code [9].

In such a case every locus on the genotype is functional — literally in that a change of a bit normally changes the value of the function that is being optimised. Fitness functions typically used in testing GAs (e.g. most of the De Jong test suite [9]) fall into this category. A fitness landscape can be visualised for such functions. In the 2-dimensional version ($n = 2$) the landscape consists of one or more hills in a bounded region of terrain where the x and y axes correspond to the two variables. The varying height of the landscape corresponds to the fitness, i.e. the value of the function for corresponding values of the two variables. Optimisation involves finding the highest hills.

Such a landscape can be generalised in two ways: firstly by extending the dimensionality to values of $n > 2$; secondly by recognising that where variables are each encoded discretely by m bits on the genotype, there are precisely 2^{mn}

genotypes, corresponding to 2^{mn} corners of a hypercube, each of which can be allocated a fitness. This hypercube visualisation recognises that genotype space is a graph or lattice of connected points; one needs a mathematician's abstract attitude to high dimensional spaces to visualise this in more than 3 dimensions. One of the messages of this paper is that one should be cautious in extending intuitions from low dimensions too readily to these high dimensional lattices where they may no longer be valid. Since a change of any one bit — a move from one corner of the hypercube to a neighbouring corner — results in a change of fitness, we call this a *non-neutral* landscape. The well-known N-K fitness landscapes [19] fit into this category.

A genetically converged population is represented by a number of nearby (in Hamming distance) points on the lattice given by the corners of the hypercube. Such a population under selection, reproduction, recombination and mutation will soon move upwards on a non-neutral fitness landscape until it arrives at a local optimum. Only if selective forces are low, or mutation rates small, will this be a protracted process.

So in these circumstances of non-neutral fitness landscapes the conventional expectation is generally correct; convergence in both senses does indeed happen at much the same time. The population converges on a local optimum, perhaps not the global optimum, and gets stuck there. *However*, much of artificial evolution, including specifically hardware evolution, will not fit this pattern of a non-neutral fitness landscape.

5 Neutrality

It is sometimes useful to borrow (and adapt) the notion of a *phenotype* from biology; in artificial evolution the phenotype is the instantiated design that a given genotype encodes — the end-product. It is open to the experimenter to choose the level of description in terms of which phenotypes are defined. For instance in the context of the FPGA hardware described here, phenotypes can be described in terms of (a) chip configuration or (b) input/output behaviour or (c) fitness score. Under options (a) or (b), two genotypes may be of equal fitness either because they encode the same phenotype, or because they encode two different phenotypes which happen to be equally fit. What counts as a change of phenotype under one description may be identical under a different description.

In this paper we are here taking option (c), the fitness score, and hence we are ignoring for present purposes changes of chip configuration which do not alter the fitness. In this context it should be recalled that under conventional conditions of population genetics, in a population of size N fitness differences between individuals that make a change in their expected number of offspring of the order of $1/N$ or less are effectively neutral (i.e. the differences are 'invisible' to selective pressure) [5, 20].

A neutral network of a fitness landscape is defined as a set of connected points of equivalent fitness, each representing a separate genotype; here *connected* means that there exists a path of single (neutral) mutations which can traverse the network between any two points on it without affecting fitness.

If a genetic encoding has a number of completely redundant loci, which are never used to determine the phenotype under any circumstance, then such loci automatically generate neutral networks. If a non-neutral genetic encoding (one which generates a non-neutral fitness landscape), with binary genotypes of length n, is modified by the addition of g extra redundant loci, then each phenotype will now be represented by 2^g points in genotype space instead of just one. These points will form a connected neutral network. However nothing will have been gained by this exercise — we shall term this type of redundancy *useless junk*.

In contrast, it is possible to add redundancy, and generate neutral networks, so as to transform the evolvability of a fitness landscape. One example [18] can transform a fitness landscape of any degree of ruggedness to a different, but related, landscape with a single global optimum; there will no longer be any local optima at which to get trapped. To do this an original encoding using n loci is transformed into one using $(1 + 2n)$. The first 'deciding' bit determines which of the following two groups of n bits is to be considered or ignored; the chosen one of these two groups is then interpreted according to the original encoding. Since for any given genotype one group of n bits is currently not being expressed, neutral mutations are possible within this group until the position of the global optimum is encoded (but not yet expressed) therein. A single mutation of the first deciding bit then generates the global optimum, without any intermediate points of lower value having been encountered.

This example is of theoretical rather than practical interest; the neutral, unselected search within the unexpressed group is no better than random search. So a landscape on which one can get trapped in a local optimum has been transformed into one in which one will never get trapped — but it will take too long (through random search) to find the goal. There are parallels between this impractical scheme and the practical operation of gene duplication plus mutation, which may be a powerful factor in natural evolution [22]. The single deciding bit acts as a form of gene switch, a concept which can be developed for generating neutral networks in artificial evolution.

This theoretical example introduces the notion of *potentially useful junk*.[2] This refers to loci on a genotype which *within the current context* of the rest of the genotype are functionless (mutations make no difference), but given different values elsewhere on the genotype they may well become significant.

In the evolved hardware which is the subject of this paper, it can be seen that the encoding allows for plenty of potentially useful junk, and consequently many neutral networks. The junk parts of the genotype where the potential for future usefulness is highest code for the periphery of the functional part of the FPGA, as shown in Figure 2. The bigger this periphery the more potential there may be. In the current experiments the only connections allowed were between neighbouring cells in the 2-D layout, but the chip does have the potential

[2] "Garbage you throw out; junk is what you store in the attic in case it might be useful one day" — attributed to S. Brenner, with reference to junk DNA.

for 'Magic wires' and 'fastlanes' which connect non-neighbouring cells, and in effect give a higher dimension topology for the chip than the simple 2-D layout. Use of these would tend to create a larger periphery to the functional part of the chip, with potentially useful effects on evolutionary pathways. With the direct encoding used in this experiment this would also possibly introduce new epistatic interactions between loci far apart on the genotype; whether or not this is desirable is currently not known.

6 Is it likely that junk is useful?

Neutral networks may be of use if they allow escape from what would otherwise be a local optimum, and this to happen within a reasonable timescale which must be much less than that of random search. The place of such networks is discussed in the context of RNA evolution in [17]. An analysis of transitions between one neutral network and the next is given in [8]. It appears that neutral networks do indeed have a beneficial effect within RNA fitness landscapes, in the sense that higher fitnesses are thereby achieved. In artificial evolution neutral networks have generally been ignored, an exception being [15].

Is this beneficial effect likely to hold true for landscapes with neutral networks such as those of hardware evolution? It is certainly not true of *all* landscapes with neutral networks; for instance the examples above with useless junk, and those with a single global optimum only reachable through random search. But there is reason to hope that much junk is potentially useful junk — though ultimately this depends on the underlying problem and on the method of encoding on the genotype. Where potentially useful junk can be generated by a suitable choice of encoding, this should be encouraged.

One relatively simple way to encourage potentially useful junk in hardware evolution is to make available many more components than are actually required at the end of the day. This was the case here, and the following analysis lends support to the hypothesis that this junk may well have been useful. Once a high fitness feasible design has been achieved, if this uses most of the available components then few mutations will leave it unaffected. If other regions of the associated neutral network exist where phenotypes are unaffected by a larger number of mutations (corresponding to a plateau rather than a ridge in the fitness landscape) then selection will tend to move the population as a whole towards such areas [3, 16, 25]. This can imply a tendency for designs that achieve the same functionality with less components to be favoured over those with more [26]; nevertheless the availability of more components may well make intermediate stages of evolution much easier by providing useful junk.

7 Genotype usage through evolutionary history

Those loci on genotypes that are currently under selection will tend to be conserved within the population at the selected values, over many generations. Other

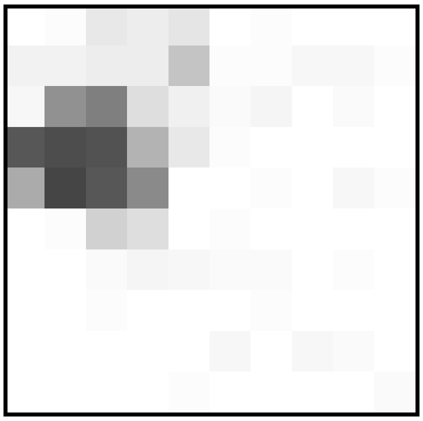

Fig. 4. For each block of 18 bits on the genotype (corresponding to one of the 10×10 cells on the FPGA), the amount of variation is calculated in the population between generations 1000 and 5000. This is plotted on a similar layout to that of Figure 2, with places of least variation darkest.

loci that are currently junk will not be so constrained; one should however be aware of the vagaries of genetic drift in small populations. Genetic drift refers to the consequences of sampling error in reproduction in small populations even when a locus is not under directed selection. The proportions of an allele at a locus will vary stochastically, or drift, until a state of all 0s or all 1s is reached; this state will generally be maintained in the medium term, until it is dislodged by mutation. With the current parameter values, 'medium term' means hundreds of generations; to see loci that are not under selection being dislodged by mutation, we must view the history over thousands of generations.

We can plot variance in alleles at each of the 1800 loci in the population between generations 1000 and 5000. This can be averaged over each block of 18 bits, which corresponds to the configuration of a single cell on the FPGA with the direct encoding used. In Figure 4 this is displayed graphically in the same layout as in Figure 2. It can be seen that there is a strong correspondence between the darker areas representing least variance, and the functional parts of the chip, as shown in Figure 2. Analysis of the genetic variation can thus be a useful tool in helping to predict which parts of the chip may be pruned away as redundant. The fact that this picture contains a lot of grey, rather than simply black and white, suggests that there has indeed been much exploration of neutral networks.

8 Saga through genotype space

The converged population is not stuck, but moves through genotype space; when it is not climbing a fitness slope it will be drifting along a neutral network. Whereas drift at a single locus may soon reach an absorbing barrier (all 0s or all

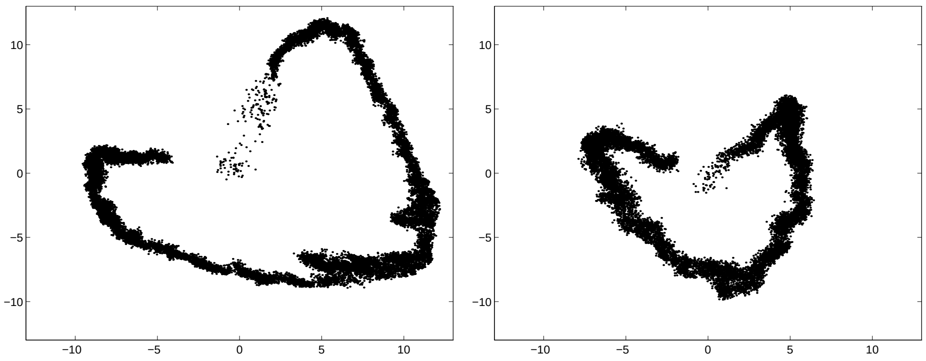

Fig. 5. The 'Saga' of the population starting from a scattered initial generation in the centre, and then in each case proceeding clockwise with a converged population; on the left data from the hardware evolution, on the right a comparison run with fitnesses randomly allocated. Every 10th generation all 50 genotypes in the population are projected onto 1st and 2nd Principal Components (PCs) derived from the population movement through genotype space over 5220 generations. The PCs are different on left and right, but the same scale has been used.

1s), neutral networks are frequently so enormous that in practice one can drift interminably. One way of visualising this movement is by projecting the 1800-dimensional genotype space onto just 2 dimensions. We choose the First and Second Principal Components of the movement of the genotype 'centroid' of the population through the 5220 generations to define the particular projection; this automatically gives a maximum spread to the displayed pathway. In Figure 5 on the left is shown every 10th generation plotted with this projection.

No special significance should be given to the gross shape of the pathway; any process of random drift or 'drunkard's walk' will give a somewhat similar path. In Section 2 we mentioned a baseline comparison where the identical GA was used, but allocating fitnesses at random without reference to the task. The pathway for this run is shown on the right.

The same scale is used on both sides of Figure 5. The population on the left is generally more converged, and moves faster, than the one on the right. In Figure 6 the speed of movement (through genotype space) of the centroids of each population is plotted, and can be seen to be some 3 times faster under selection than in the random case.

In Figure 5 on the left, there are two periods when the population appears to spread out; at "4 o'clock" and again at "5 o'clock". The latter period, leading up to the transition around generation 2660 (visible in the mean fitnesses in Figure 1), shows the population for any one generation spread out into a cigar shape under this projection. In Figure 7 on the left is shown the elite member of generation 2550 and the rest of the population.

This projection has been calculated so as to emphasise the spread of geno-

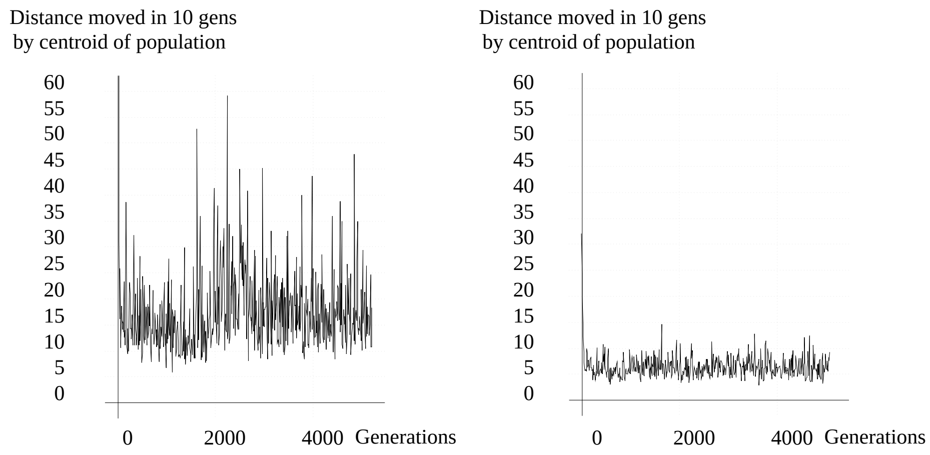

Fig. 6. Plots of how fast the 'centroids' of the populations shown in Figure 5 move; on the left with evolving hardware, on the right with randomly allocated fitnesses.

types from the population history or saga. The corollary of this is that genotypes that are in no way associated with this saga, such as random genotypes, are likely to have only very small components in the direction of these axes, and will appear near the origin or centre of this projection (as indeed the initial random populations do in Figure 5).

In general the elite member of any snapshot is always furthest from the origin of this projection. Non-elite members of a population tend to be mutations derived from the elite of earlier generations; mutations tend to make the genotype 'more random' — have less association with the path of the population — and hence are nearer the origin than the elite.[3] This can be seen on the right hand side of Figure 7, where the 'comet's tail' of mutants streams away from the elite member towards the origin; these mutants were generated by taking 50 copies of the elite, and applying mutations to each at rates varying linearly between 0 and 6.12 expected mutations per 1800 bits, to produce a spread roughly comparable with the left hand side. On the left, with the data from hardware evolution, the comet's tail points off to one side of the origin, indicating that these genotypes are not solely the product of random mutation, but rather of mutation plus selection. One can conclude that the population is spread out along high value ridges in the fitness landscape, which is where even fitter regions may be found. This corresponds to the picture put forward by Eigen et. al. in [4], pages 6886–6887:

[3] This follows from the fact that in high-dimensional spaces two random vectors are almost always nearly orthogonal; the projection of one onto the other (dot product) can be expected to be small.

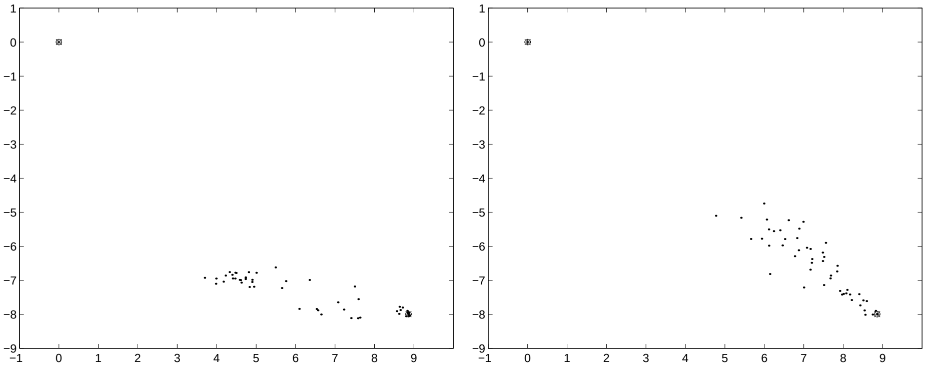

Fig. 7. The same projection is used here as in Figure 5, except it is scaled up and the origin (marked) is now at the top left. In the left figure all 50 genotypes of generation 2500 of hardware evolution are plotted; the elite member is marked at bottom right. On the right the same elite member is shown with 49 other genotypes mutated at random without selection; it can be seen that the population forms a 'comet's tail' pointing towards the origin in this latter case. Here the same PCs are used for the projections on the left and the right.

> "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."

9 A picture of the landscape

Some indication of the nature of the fitness landscape can be found by plotting the fitnesses of each snapshot taken of the population during evolution. In Figure 8 the fitnesses of each individual in the population are ranged in order within each snapshot, and then ranged alongside each other to cover the 5220 generations (with one snapshot every 10th generation). Necessarily this landscape does not show *all* of the fitness terrain around the current population, but only that part actually sampled (with noise) by genotypes generated through evolution.

Throughout the run there is a small part of the population, usually around 10 out of 50, which has zero or near-zero fitness. After generation 2660 there is a high plateau on which most of the population lies; this is an indication of how much neutral mutation is possible at this stage. For a period leading up to this plateau there is a narrow ridge which holds the current elite, with almost all

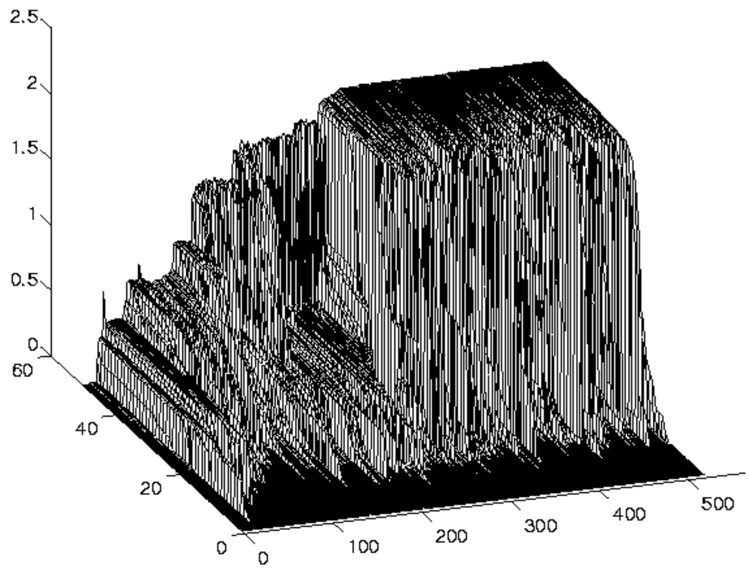

Fig. 8. The fitness landscape, as sampled by the population. Over 500 snapshots are displayed from left to right (one every 10 generations). Within each such snapshot the 50 members of the population are ranked (from back to front of the figure) according to fitness, which is plotted on the vertical scale. After the initial stages a narrow ridge leads up to a flat plateau which starts around snapshot 266, which is generation 2660.

the population considerably less fit. This period corresponds to the increase in genetic diversity in the population (Figures 3 and 5). If the GA had not used the strategy of elitism, this type of ridge might not have occurred; this speculation has not been checked by experiment.

10 Use of SAGA

Normal GAs, indeed most other EAs, are not designed to be used with the genetically converged populations that we have seen here. The Species Adaptation Genetic Algorithm, or SAGA [12, 10, 13], is designed expressly for this purpose. The differences from normal GAs are subtle but significant, and the core requirements of SAGA have been used in the hardware evolution described here.

These can be briefly summarised. Evolution is directed by selection exploiting differences in fitness causes by variations in the genetic make-up of the population. Conventional GAs and GP usually assume that all or nearly all of the variation is that of the initial random population, and hence evolution will cease when this initial variation is exhausted; mutation is thus considered a background genetic operator, and recombination is usually assumed to be the primary operator. In SAGA the reverse is true; of these two genetic operators, mutation is primary, and recombination (though useful) is secondary.

The effectiveness of evolutionary search in a converged species is largely determined by the balance between selection and mutation. Zero mutation leads to stasis, and too much mutation for the selective pressure leads to loss of whatever useful genetic information may have accumulated previously in the genotypes (the 'error catastrophe'); an optimum must lie between these two extremes, and in fact is the maximum rate which still lies below the error catastrophe. In [4] it is shown that in an asexual population with a large population size, on a fitness landscape consisting of a single peak on an otherwise flat plain, the error catastrophe occurs at a rate of $m = \ln(\sigma)$ expected mutations per genotype; where σ is the *superiority* parameter of the master sequence — the factor by which a fittest member on the peak outbreeds the average members on the plain.

For the rank-based selection scheme used here (which includes elitism), $\ln(\sigma) \sim 1$; other selection schemes with significantly different selection pressures need adjustment accordingly. Adding recombination to an asexual population, and reducing population size to the small figures we use such as 50, makes relatively little difference to the impact of the error catastrophe on the effectiveness of evolutionary search [13]. The change from the rugged single-peak landscape to realistic fitness landscapes may make more difference, but always such as to increase the critical error rate.

What *does* make a significant difference, however, is the presence of redundancy or junk in the genotype. If the target mutation rate is, as in our case, 1 mutation per genotype on the assumption of no redundancy, in the presence of junk this should be increased so as to give an expected 1 mutation per *non-redundant* part of the genotype. Usually one does not know in advance what proportion will be junk — it may alter during evolution — but estimates can be checked as evolution progresses. In this case the mutation rate used of 2.7 bits mutated per genotype would be safe (i.e. less than the error catastrophe) if at least 63% of the genotype was redundant; results suggest that this figure was a reasonable one to use. For further details of SAGA [12, 10, 13] should be consulted. A fundamental aspect of the SAGA approach is that of incremental evolution: after success at some level of complexity future evolution at more complex related tasks can continue from the current converged population, rather than starting again with a random population.

11 Conclusion

The picture of artificial evolution presented here differs significantly from that in the main body of EA literature. We have demonstrated with an example of evolution of the hardware configuration on a real silicon chip that fitness improvement can indeed take place within a genetically converged population. With this different approach there is no need to worry about premature convergence, and thus small population sizes may be perfectly acceptable.

An attempt has been made to promote a different perspective on fitness landscapes. Whereas one *can* create such landscapes with isolated hills which allow a population to get trapped at local optima, it is suggested that landscapes associated with many real problems are *not* of this nature. If there is redundancy of the right kind — 'potentially useful junk' — then neutral networks percolating through genotype space may eliminate most local optima (perhaps all that are likely to be encountered) except for global optima. Though some neutral networks may not be useful, it may be possible to encourage the presence of the practical kind by allowing for many more components to be available for hardware evolution than will be necessary in the final design.

There is a lot more to evolution than meets the eye, and naive models and metaphors may lead to poor decisions in the design of evolutionary algorithms, or prejudice against reasonable decisions. In the context of much contemporary EA practice, the use of a small population of size 50, with genotypes of length 1800 bits, continued for 5000 generations with a genetically converged population on a hard real-world problem, would seem to many to be folly. With the different perspective of SAGA, and consideration of the role of junk and neutral networks, it seems more plausible. The actual result achieved, on a real silicon chip, supports the choice of method.

References

1. H. Asoh and H. Muehlenbein. On the mean convergence time of evolutionary algorithms without selection and mutation. In H.-P. Schwefel Y. Davidor and R. Männer, editors, *Parallel Problem Solving from Nature (PPSN III), Lecture Notes in Computer Science 866*, pages 88–97. Springer-Verlag, 1994.
2. N. L. Cramer. A representation for the adaptive generation of simple sequential programs. In J.J. Grefenstette, editor, *Proceedings of an International Conference on Genetic Algorithms and Their Applications*, Hillsdale NJ, 1985. Lawrence Erlbaum Associates.
3. M. Eigen. New concepts for dealing with the evolution of nucleic acids. In *Cold Spring Harbor Symposia on Quantitative Biology, vol. LII*, 1987.
4. M. Eigen, J. McCaskill, and P. Schuster. Molecular quasi-species. *Journal of Physical Chemistry*, 92:6881–6891, 1988.
5. D.S. Falconer. *Introduction to Quantitative Genetics, 3rd edition.* Longman, 1989.
6. D. B. Fogel. *System Identification through Simulated Evolution.* Ginn Press, Needham, MA, 1991.
7. L.J. Fogel, A.J. Owens, and M.J. Walsh. *Artificial Intelligence through Simulated Evolution.* John Wiley, New York, 1966.
8. C.V. Forst, C. Reidys, and J. Weber. Evolutionary dynamics and optimization: Neutral networks as model-landscapes for RNA secondary-structure folding-landscapes. In F. Moran, A. Moreno, J.J. Merelo, and P. Cachon, editors, *Advances in Artificial Life: Proceedings of the Third European Conference on Artificial Life (ECAL95). Lecture Notes in Artificial Intelligence 929*, pages 128–147. Springer Verlag, 1995.
9. D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, Reading MA, 1989.

10. I. Harvey. Evolutionary robotics and SAGA: the case for hill crawling and tournament selection. In C. Langton, editor, *Artificial Life III, Santa Fe Institute Studies in the Sciences of Complexity, Proc. Vol. XVI*, pages 299–326. Addison Wesley, 1993.
11. I. Harvey, P. Husbands, and D. T. Cliff. Genetic convergence in a species of evolved robot control architectures. In S. Forrest, editor, *Genetic Algorithms: Proceedings of Fifth Intl. Conference*, page 636, San Mateo CA, 1993. Morgan Kaufmann.
12. I. Harvey. Species adaptation genetic algorithms: A basis for a continuing SAGA. In F. J. Varela and P. Bourgine, editors, *Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, pages 346–354. MIT Press/Bradford Books, Cambridge, MA, 1992.
13. I.R. Harvey. *The Artificial Evolution of Behaviour.* PhD thesis, COGS, University of Sussex. Also available via www on http://www.cogs.susx.ac.uk/users/inmanh/, 1995.
14. J. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, USA, 1975.
15. T. Hoshino and M. Tsuchida. Manifestation of neutral genes in evolving robot navigation. Presented at Artificial Life V, Nara, Japan, May 1996.
16. M. Huynen and P. Hogeweg. Pattern generation in molecular evolution: Exploitation of the variation in RNA landscapes. *J. Mol. Evol.*, 39(7):1–79, 1994.
17. M.A. Huynen, P.F. Stadler, and W. Fontana. Smoothness within ruggedness: The role of neutrality in adaptation. *Proc. Natl. Acad. Sci. USA*, 93:397–401, January 1996.
18. N. Jakobi. Encoding scheme issues for open-ended artificial evolution. In Voigt, H.-M., et al. (eds.), *Parallel Problem Solving From Nature IV: Proc. of the Int. Conf. on Evolutionary Computation*, Vol. 1141 of *LNCS*. Heidelberg: Springer-Verlag. Forthcoming.
19. S. A. Kauffman. *Origins of Order: Self-Organization and Selection in Evolution.* Oxford University Press, 1993.
20. M. Kimura. *The Neutral Theory of Molecular Evolution.* Cambridge University Press, 1983.
21. J. R. Koza. *Genetic Programming.* MIT Press/Bradford Books, Cambridge MA, 1992.
22. S. Ohno. *Evolution by Gene Duplication.* Allen and Unwin, London, 1970.
23. I. Rechenberg. *Evolutionsstrategie: Optimierung techniser Systeme nach Prinzipien der biologischen Evolution.* Frommann-Holzboog Verlag, Stuttgart, 1973.
24. H.-P. Schwefel. *Numerical Optimization of Computer Models.* John Wiley, Chichester, U.K., 1981.
25. A. Thompson. Evolutionary techniques for fault tolerance. In *Proceedings of the UKACC International Conference on Control 1996 (CONTROL'96)*, pages 693–698. IEE Conference Publication Number 427, 1996.
26. A. Thompson. An evolved circuit, intrinsic in silicon, entwined with physics. In this volume: *Proceedings of The First International Conference on Evolvable Systems: From Biology to Hardware, 1996 (ICES96)*, Springer-Verlag.