Artificial Evolution for Real Problems

INMAN HARVEY
CCNR, COGS
University of Sussex
Brighton BN1 9QH, UK
Tel: +44 1273 678754, Fax: +44 1273 671320
Email: inmanh@cogs.susx.ac.uk

1 Introduction

Humans are naturally evolved creatures, and the selection criteria under which our ancestors were judged did not include the ability to design complex systems — in fact, we are not very good at it. A common and useful trick to overcome our shortcomings is that of *Divide and Conquer*; a complex problem is decomposed into separate, less daunting, sub-problems. This can be seen in the division of labour which the industrial revolution brought about in the workplace. Likewise in good computer programming practice, the overall problem is broken down into smaller pieces that can each be tackled by a separate function; these functions should be carefully fenced off from each other, interacting only through the passing of values in a well-defined manner.

Only the relatively easy problems can be handled successfully using this *Divide and Conquer* trick — the problems that we have had some success with. But where interactions within a complex system are so many, or so little understood, that we cannot understand how to divide it into smaller comprehensible pieces, then we will not be able to control it for our purposes. It is for these kinds of problems that artificial evolution is put forward as a means to develop complex systems that work even when they are too complex to comprehend.

I will first introduce Genetic Algorithms (GAs) as the best-known appli-

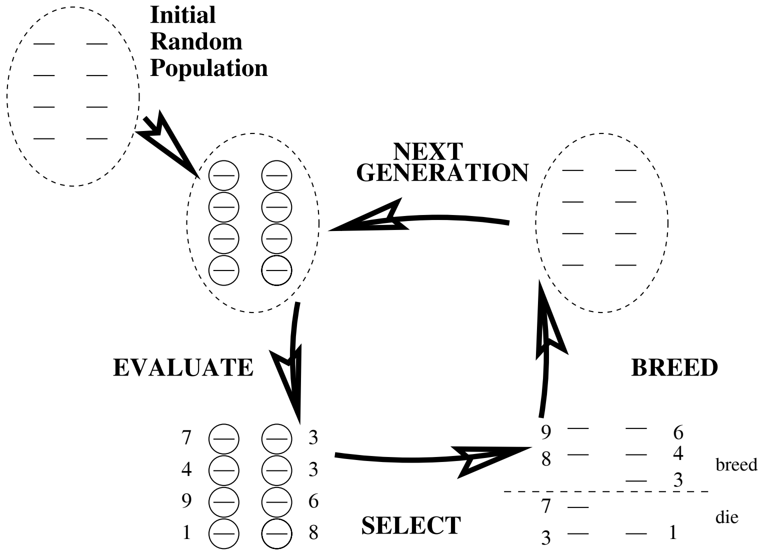

Figure 1: The Genetic Algorithm Cycle. The lines represent genotypes or linear strings of characters. The circles containing genotypes represent the corresponding trial solutions to the problem, which can be evaluated.

cation of ideas from natural Darwinian evolution to engineering applications. I shall then point to areas where standard GAs used for optimisation are not the right tools, and show how GAs can be extended to deal with incremental evolution. Some applications in evolutionary robotics, evolutionary hardware and combinatorial chemistry will be discussed briefly, and then I will survey very recent developments which are the topics of current research in the field.

2 Genetic Algorithms

GAs are generally used as a search technique to optimise a parameter set or a design for a system; where different trial parameter sets or trial designs can be evaluated against some particular criterion. So the parameters may be the dimensions of the different parts of an aircraft wing which must balance high aerodynamic lift and high structural strength against low weight and ease of manufacture; or the design may be that of a control system for a robot which has to generate some desired behaviour; or of a molecule to be synthesised for pharmaceutical purposes, to maximise its binding properties with some other target molecule; or the configuration of a silicon chip to be used as an analogue circuit. In every case some 'genetic encoding' must be

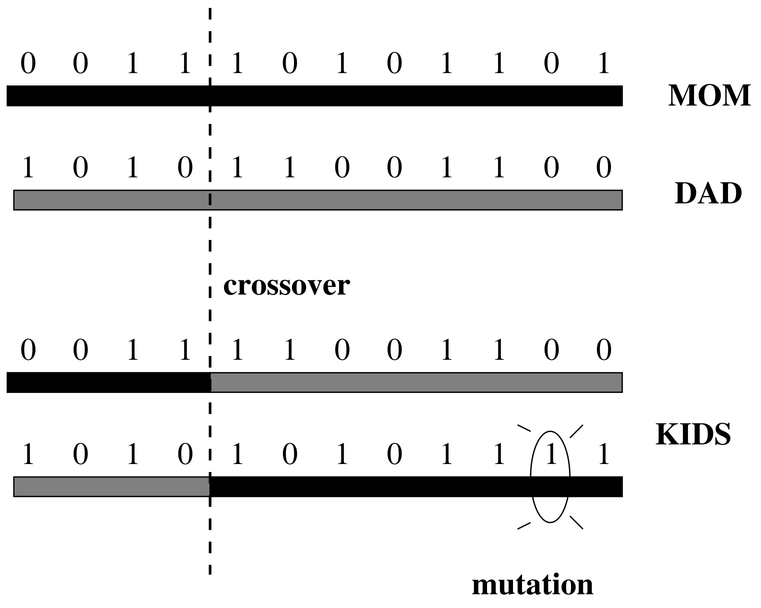

Figure 2: A random crossover point is chosen on the two parent genotypes, and offspring genotypes derived from them. Additional mutations can change individual characters on the resultant genotypes.

devised to suit the particular problem, such that any possible trial parameter set or design can be written down in the form of a string of characters which can be considered as a genotype of artificial DNA.

The GA then uses a *population* of such genotypes; for example 100. The initial population may be 100 randomly generated genotypes, each of which encodes a trial solution to the problem. Each solution can be evaluated in turn (or possibly in parallel) and given a score, its 'fitness'. For randomly-generated genotypes these fitnesses will probably be extremely low, but nevertheless through chance some will be higher than others. As indicated in Figure 1 the fitter members of the population are chosen to be 'parents' of the next generation, whose genotypes will be derived from their parents.

Selection of the fitter parents is generally done in one of many probabilistic methods [2] which still give a small chance even to the less fit; though one option of truncation selection is to rank the population in order of fitness and simply choose the first n fittest as parents. Pairs of parents are picked from this pool, and from these parents offspring are derived in pairs as sketched in Figure 2. Recombination means that each offspring inherits some genetic material from each parent, and mutations add some variety.

The process of generating new offspring is continued until a number equal to the population size has been reached. this forms the new generation; the previous generation is discarded and replaced by the new one. The cycle starts again (Figure 1) with the evaluation on the task of each new trial solution as specified by the new genotypes.

If the genetic encoding is appropriate to the problem, then ideally the succeeding generations under the selection pressure should become fitter and fitter under the evaluation scheme, until near-optimal solutions to the particular problem are found. There are a number of factors that may prevent a GA working in this ideal fashion, however. The process of searching for a fit solution may take too long to be practicable, for instance, or a local optimum may be found which prevents any further progress. There are a number of ways in which the standard GA can be varied, which may affect these issues.

1. The population size can be increased or decreased; unless the population can be evaluated in parallel this will affect the length of a GA run.

2. The Generational GA discussed here can be replaced with a Steady State one, where only one or two members of the population are replaced at each stage.

3. A variety of Selection schemes are possible, and these can increase or decrease the selective pressure.

4. There alternatives to the single-point crossover form of Recombination; for instance two-point crossover, or even uniform crossover where each 'locus' or character on an offspring genotype has an independent 50% probability of being inherited from either parent.

5. Mutation rates can be altered.

6. A 'Distributed GA' can be used, as in Figure 3.

This latter type of GA requires the use of a hypothetical geographical space in which each member of the population is notionally located in a grid cell. The GA operations of selection, recombination and reproduction happen within the local area around any given member (the grid is considered as toroidal, so that top and bottom edges meet, left and right edges meet).

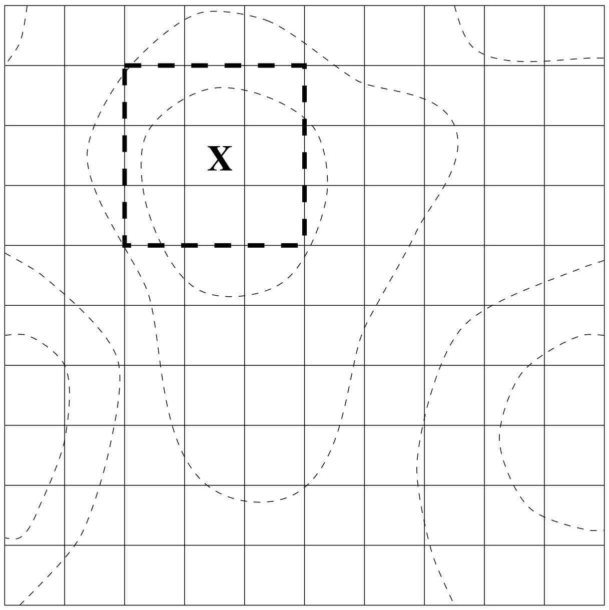

Figure 3: A distributed GA. Each member of the population is allocated a different square of the grid. If member X is selected as a parent, then the second parent for recombination is chosen from its immediate neighbourhood, for example the 3×3 square. Resulting offspring are also placed locally on this grid, so that neighbourhoods contain closely related members.

This allows for the maintenance over several generations of different subpopulations which focus on different areas of the search-space; in effect different sub-species, which still have the possibility of recombining where they are neighbours. One effect of such a distributed GA is, by maintaining variation in the population, to make it more unlikely to get trapped in a local optimum.

3 SAGA and Incremental Evolution

So far I have described the standard GA which is widely used for optimisation. It can for instance be used as an alternative to Simulated Annealing, with which it bears some similarities. A main distinguishing feature of GAs is that they are a form of *population search*. One abstract way of visualising how they operate is to think in terms of *genotype space*, the high-dimensional space which contains as separate points each possible genotype. Each such point can in theory be allocated a score, the evaluation of the trial solution which the corresponding genotype represents; but typically there are far too many points for an exhaustive search.

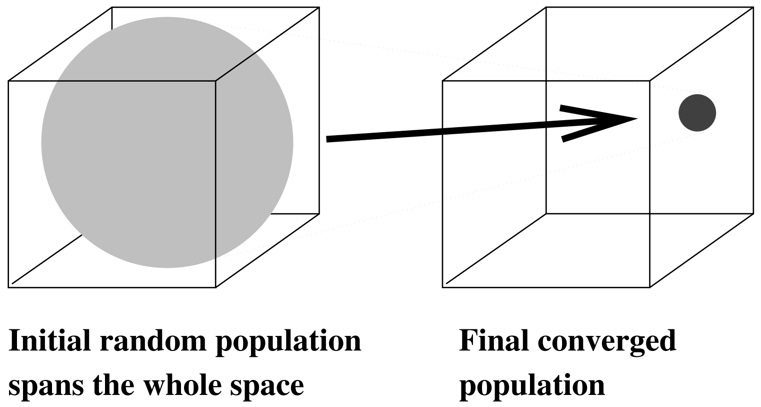
Initial random population spans the whole space
Final converged population

Figure 4: A two-dimensional sketch of a multi-dimensional genotype search space. An initial population samples widely through the space, and through successive generations of a GA cycle the population becomes focused in a (near-)optimal region.

The theoretical background to GAs such as the Schema Theorem [2] implies that when an initial random population is widely scattered across genotype space, as in Figure 4, the successive application of selection, recombination and mutation is an efficient method of using the information gained from the relatively small number of evaluated points to focus the population in onto a region of high fitness, where good solutions to the problem are to be found. This corresponds in standard GAs to the concept of *convergence* to a good solution.

From now on this paper departs from standard GA orthodoxy, as from now on I shall concentrate on SAGA (Species Adaptation Genetic Algorithms), which uses rather different assumptions [9, 8, 3]. In contrast to the previous picture, in SAGA the population is always genetically fairly converged — the genotypes of different members of the population are rather similar to each other. Whereas conventional GAs consider genetic convergence to the end of an evolutionary run, in SAGA it is only the beginning (Figure 5).

Artificial evolution with genetically converged populations, for which the SAGA approach is tailored, occurs under the following circumstances:

1. Incremental evolution, when the problem itself changes over time; for instance a sequence of related problems of increasing complexity may be tackled by a continuing evolutionary algorithm.

2. This may include cases where the genotype length may change in the

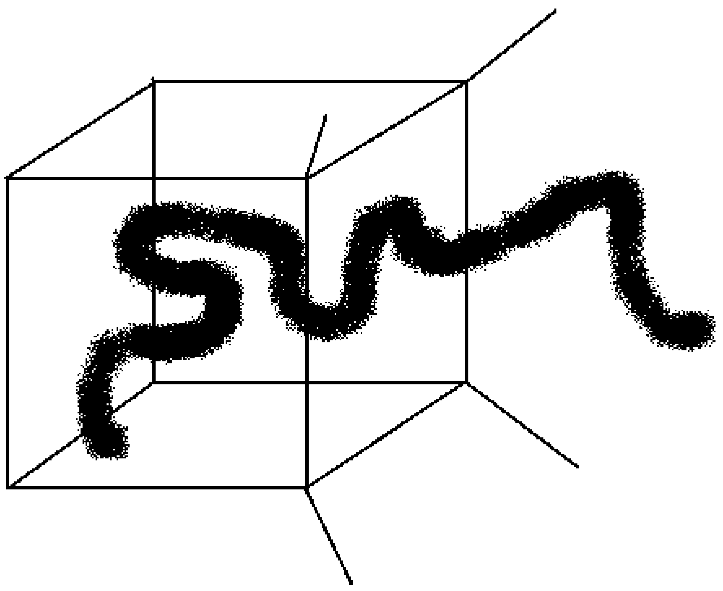

Figure 5: A sketch of the movement of a converged population through genotype space under SAGA. The 2-dimensional picture represents a high-dimensional space, which may even be increasing in dimensionality with successive generations.

course of evolution, particularly where increasingly complex solutions require increasing genotypes in order to specify them.

3. It is becoming increasingly recognised that even in standard GA optimisation problems, an initially random population typically becomes genetically converged after very few generations, and hence SAGA principles are relevant.

The first case has been studied at length in the context of Evolutionary Robotics [6, 4]. Artificial evolution can be used to evolve robots to perform a sequence of increasingly complex tasks. The genotypes encode the architecture of the robot control systems (Figure 6), plus possibly also some aspects of their sensory morphology. In these examples the robot tasks were navigational tasks towards target objects which were to be distinguished by vision. A real robot and real vision were used, avoiding some of the simulation problems to be discussed later.

Initially the target object to be identified and approached was extremely simple — a long white wall contrasted with a dark background. The population of robot genotypes was evaluated for robot prowess at this particular

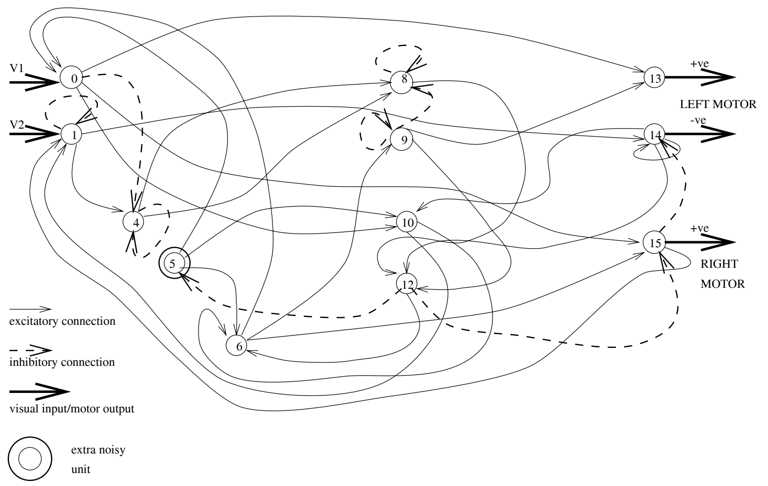

Figure 6: A genetically specified control architecture for a robot. This is a form of recurrent neural net, connecting robot sensors with robot actuators via intermediate nodes. This net implements a control system which produces successful robot behaviour at a specific navigational task, but as it was evolved rather than designed by a human, the rationale for its operation is not immediately clear.

target for a number of generations, until success was achieved. Then the target was made more difficult to distinguish, and evolution continued with the same population under these new conditions. A succession of such changes of target took place, with a genetically converged population.

This scenario has properties common to many engineering domains, where the specification for this year's model is a tightening up or an extension of last year's. If artificial evolution is the design methodology, then it is a waste of time to start a GA for each new specification with a random population; instead one should use the SAGA approach of adapting the population that was evolved for the previous specification.

SAGA differs from standard GAs because of the use of a genetically converged population. Recombination, whilst still relevant, has a lesser role, and mutation is probably more important as a genetic operator.

1. The converged population will almost always be (to a first approximation) centred around some current local optimum, or hill in the fitness landscape (Figure 7).

2. With too little mutation (or too high a selective pressure) no further progress will be made.

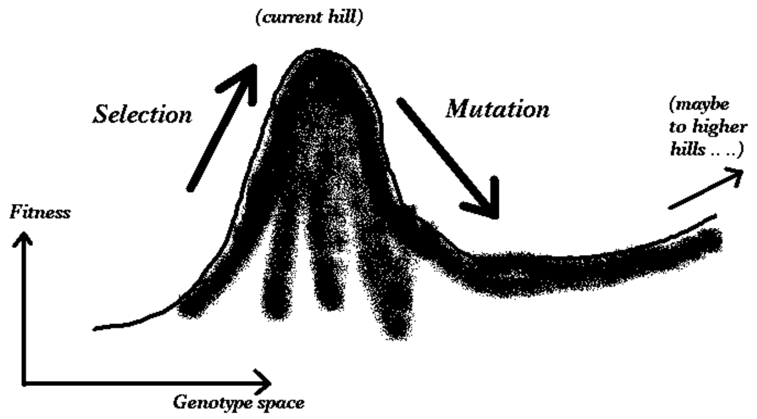

Figure 7: The balance between the forces of selection and mutation on the fitness landscape. Here the vertical dimension represents fitness, and the horizontal dimension represents all the many dimensions of genotype space. The population will move over this landscape through successive generations.

3. With too much mutation (or too little selective pressure) any useful information gained through earlier evolution will be lost.
4. With the right balance optimal search will take place along high 'ridges' in the fitness landscape which may lead to even fitter solutions.
5. To a second approximation, *neutral networks* in the fitness landscape will be significant; to be discussed in a later section.

The ideal balance between the forces of selection and mutation occurs, under normal selection scenarios, when there is of the order of 1 mutation per genotype (regardless of the length of the genotype). This assumes for the time being that any part of the genotype will have some effect if it is altered; when we come to discuss 'junk DNA' this figure will need to be adjusted.

4 Smooth Fitness Landscapes

So far the discussion has been general, not specific to any particular problem. When it comes to applying these ideas in a specific problem domain, the first question to be tackled is the choice of encoding from a linear genotype onto a trial solution of the problem, a 'phenotype'. Different choices made at this stage may well result in different fitness landscapes, and some such landscapes

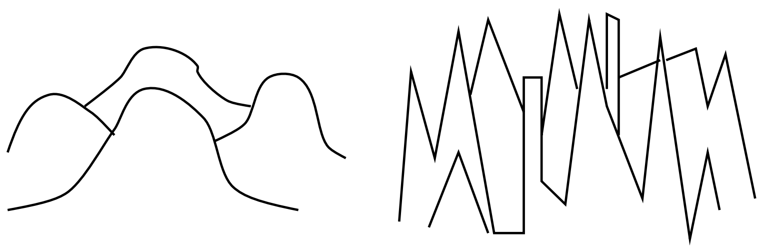

Figure 8: Evolution is more effective on smooth fitness landscapes as on the left, rather than rugged ones wherein similar genotypes typically have uncorrelated fitnesses.

are more amenable to evolution than others. Since most of evolutionary progress is made through successive mutations, then a rugged landscape in which neighbouring genotypes have fitnesses usually uncorrelated with each other will cause problems. A gently rolling landscape is better than a rugged one (Figure 8).

One lesson that can be drawn from this is that genotypes as representations of computer programs give inherently rugged fitness landscapes and hence are not suitable for artificial evolution. The equivalent of a mutation in a computer program is an arbitrary change of a single character to something else, and any computer programmer who has written in C or in assembly code knows that this is almost always fatal to an otherwise functional program. The achievements of Genetic Programming (GP) [13] do not contradict this, when one considers continual evolution of a genetically converged population. The use of GP is generally confined to a relatively small number of generations, and often a very large population. Recombination effectively mixes and matches the variation in the initial population, but this variation is rapidly exhausted in very few generations. Once this has happened, no further evolution can take place.

Hence although GP may be appropriate for some specific optimisation problems, it is not appropriate for the domains where mutation in a converged population is required, as in SAGA.

When artificial evolution is used for evolutionary robotics, then a smoother fitness landscape is achieved when there is noise within the control systems; as then mutations in the structure have effects not so far different from those of noise that is already present. This has been used with some success in the robotics work cited earlier.

When evolving hardware circuits [14] on Field Programmable Gate Arrays

(FPGAs), then treating them as analogue circuits may well result in evolution being far easier than if they were constrained to act as digital circuits. I would suggest that the efforts of most current practitioners of hardware evolution to evolve digital circuits will, in the long run, have the same limitations as such paradigms as GP; it may well be that only short-term optimisation, not long-term continued evolution, is possible. The work by Thompson [14] avoids this limitation.

5 Neutral Networks

One worry about the use of a genetically converged population in SAGA is that the population may get stuck on a local optimum in some corner of genotype space, far from any useful solutions. However there is a large class of problems for which there is good reason to believe that this is unlikely.

These are problems where the genotype encodes aspects of the phenotype (or trial solution) in a potentially multiply redundant fashion. Examples include the three domains mentioned earlier: evolutionary robotics, hardware evolution, and combinatorial chemistry. In all these cases parts of the phenotype may — or may not — be functional, depending on the context of the rest of the genotype. In the control networks used in the robotics work mentioned, individual connections or subnetworks have effects that may be masked or eliminated by the presence or absence of other connections elsewhere. In the hardware example discussed below there are comparable effects. Where a genotype is encoding the linear structure of a molecule which folds in three dimensions, the binding properties of one part of its length to a target molecule depend on whether this part is exposed on the 3-D surface, or is altered by contact with other parts of the molecule.

In all these cases there is a very-many$\rightarrow$one mapping between genotypes and any one particular phenotype, where the phenotype is the behaviour (of robot, hardware or molecule). This can result in *neutral networks* in genotype space (Figure 9), where there are connected networks of genotypes in genotype space which can be traversed through single mutations, yet which code for the same phenotype. Models of RNA landscapes show that such neutral networks can percolate throughout genotype space [10], implying that any target phenotype can almost always be found fairly nearby to any starting

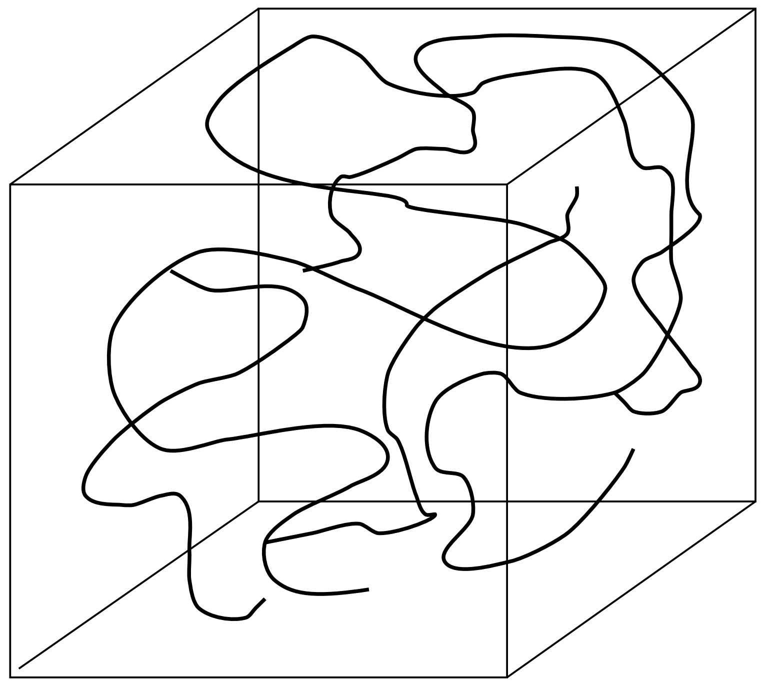

Figure 9: Neutral networks, of neighbouring genotypes which encode the same effective genotype, can under some circumstances percolate widely through multi-dimensional genotype space.

point. Hence it is difficult to get trapped in a local optimum, and much of evolutionary search is the process of a population 'searching' around the neutral network associated with the fittest phenotype reached to date. When such search happens upon another fitter neutral network, the population will 'jump' onto it, in a form of punctuated equilibrium.

This scenario shows that the simplistic picture of Figure 7 may be partially misleading for the high-dimensional spaces that genotypes actually support. There is reason to believe that such neutral networks heavily influence the manner in which evolution takes place, and algorithms such as SAGA need to take account of this.

6 Simulations

Evolution requires the evaluation of vast numbers of trial solutions to the problem being tackled. For interesting real world problems, such evaluations take time and money; indeed this is the bottleneck in a GA, as the genetic operations are trivial in comparison. It is tempting, therefore, to use simulations of the problem domain in which to evaluate trial solutions.

Simulations however must be validated; it is all too easy to evolve solutions that work in simulation yet fail to work in reality. In addition, for some problems simulations may be too computationally expensive to be practical options, and testing things in reality may be cheaper. In the context of evolutionary robotics, Jakobi has developed a theory and a methodology for deciding just how minimal a simulation can get away with being, yet still be adequate enough for evolved solutions to work when transferred to real robots [12, 11]. By carefully assessing which parts of the real world should *not* be used by the robot to influence its behaviour, and replacing these aspects with noise in the simulation, Jakobi can produce minimal simulations that can run very fast; and he has shown that these can be effective in evolving designs that work in the real world.

7 Hardware Evolution

One area where simulations have been shown to be inadequate is that of hardware evolution, when using an FPGA as an analogue device [14]. In this work genotypes of length 1800 bits specified the configuration of a section of FPGA containing 100 logic blocks, and evolution was carried out evaluating the FPGA at a signal-recognition task — the real chip, not a simulation.

A successful evolved design is shown in Figure 10. By testing the chip the functional subset of connections was determined, as shown on the right of that Figure. The shaded portions shown affected the functionality of the chip configuration if they were clamped to fixed values — yet those same shaded logic blocks were not directly connected to the functional circuit. Presumably as yet unknown side effects such as capacitance were implicated in this, but without knowing what these effects were, no simulation could model them.

As can be seen from Figure 10, although the genotype specified details of all 100 logic blocks (shown in a 10×10 grid on the left), some 60–70% of these details were redundant (corresponding to blank areas on the right). In other words some 60–70% of the genotype was redundant or 'junk DNA' in that successful genotype; mutations in such junk parts would not affect the

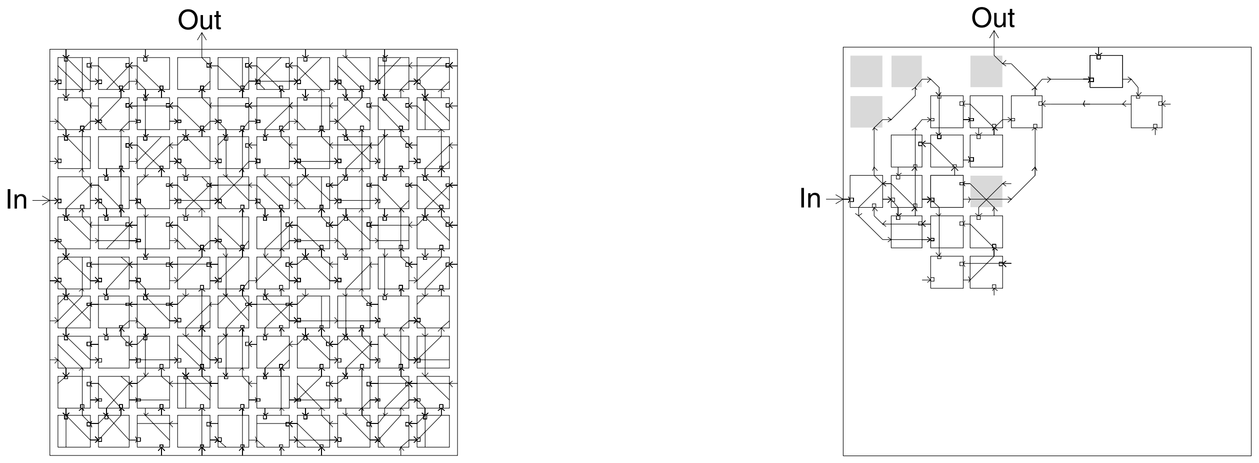

Figure 10: On the left is shown the genetically specified configuration of the FPGA after evolution for 5000 generations. On the right is shown the subset of these connections which is functional, in the sense that the hardware still functions appropriately when the rest of the chip is clamped to fixed values.

fitness of the reconfigured chip, except perhaps some of those immediately neighbouring the functional part of the circuit.

This has at least two implications. Firstly, the ideal mutation rate that balances selection pressure against mutation needs to be adjusted to take account of this, since mutations in the junk part of the genotype have no *immediate* effect. Hence in this example the mutation rate used of some 2.7 mutations per whole genotype corresponds to about one mutation per *functional, non-junk* proportion of the genotype. Secondly there can now be seen the possibility of enormous numbers of different genotypes each encoding the same phenotype — where now 'phenotype' refers not to the complete FPGA configuration, but to the *behaviour* of the chip in use. This demonstrates the existence of neutral networks in genotype space.

Neutral networks need to percolate widely through genotype space to be of assistance in aiding evolutionary progress, as in Figure 9. Where one specific region of the genotype is junk, or non-functional, under all contexts, then the neutrality conferred by this redundancy is no assistance as it will be confined to a relatively small region. What would be useful is when some regions of the genotype are junk in some contexts, but not in others, and this can be seen to be the case in this FPGA example. Some areas of the chip may be unused in the successful configuration shown, but in earlier

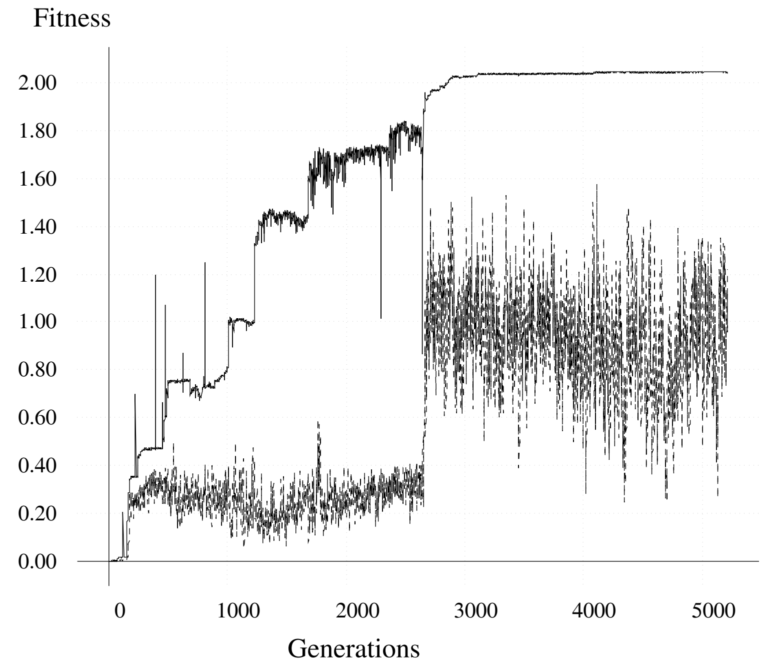

Figure 11: The maximum and mean fitnesses of the population are plotted over 5220 generations. The dramatic rise in mean fitness occurs at generation 2660.

'ancestor configurations' they would have had an effect on the chip behaviour. This implies that intermediate neutral networks (passed through during the evolutionary process) overlap and percolate widely through genotype space, thus making entrapment on local optima less likely.

8 Analysis of a Fitness Landscape

The evolutionary pathway of Thompson's experiment of [14] was analysed in detail in [7], and this will be summarised here as it sheds light on more general issues in artificial evolution.

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 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

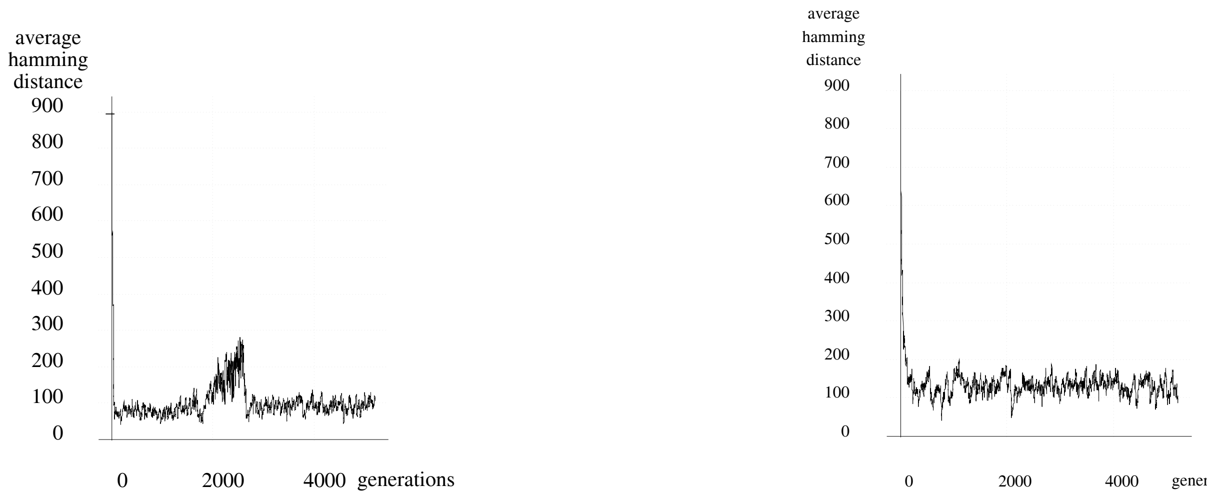

Figure 12: Plot of genetic convergence within the population (measured as average Hamming distance between pairs of genotypes) against generations. On the left, from the evolving hardware; on the right, with fitnesses randomly allocated.

median-ranked individual. Single-point crossover probability was 0.7, and the 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 11 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 the signal-discrimination task on which the chip was being evaluated.

The fact that some 2/3 of the 1800 bits were redundant in the genotype that produced the successful configuration of Figure 10 implies that 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; massive redundancy.

In Figure 12 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 11.

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 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 12. 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 [5]. For many users of GAs this is unexpected, but in fact analysis of conventional optimisation GAs will often show similar characteristics.

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 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 13 on the left is shown every 10th generation plotted with this projection. This can be compared with the abstract sketch of Figure 5.

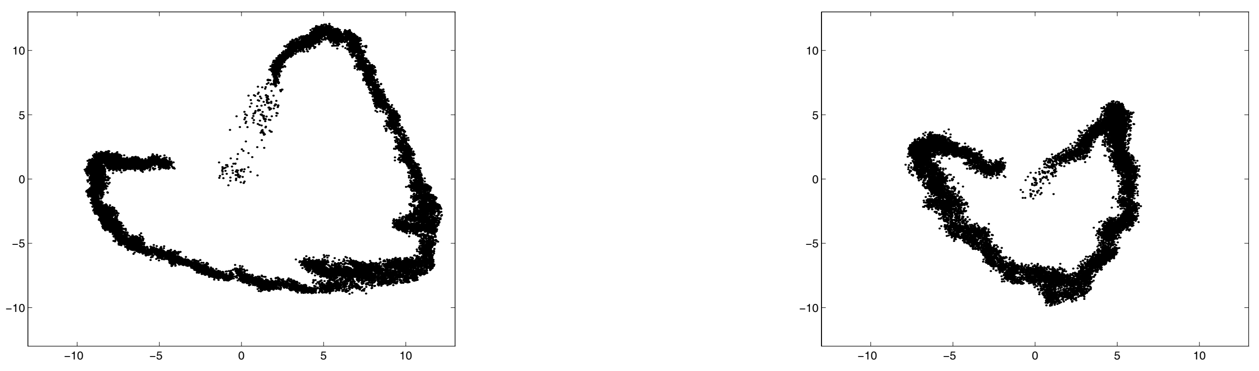

Figure 13: The 'Saga' of the population starting from a scattered initial generation in the centre, and then in each case proceeding clockwise with a converged population; on the left data from the hardware evolution, on the right a comparison run with fitnesses randomly allocated. Every 10th generation 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.

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. Above 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.

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 the population considerably less fit. This period corresponds to the increase in genetic diversity in the population (Figures 12 and 13). If the

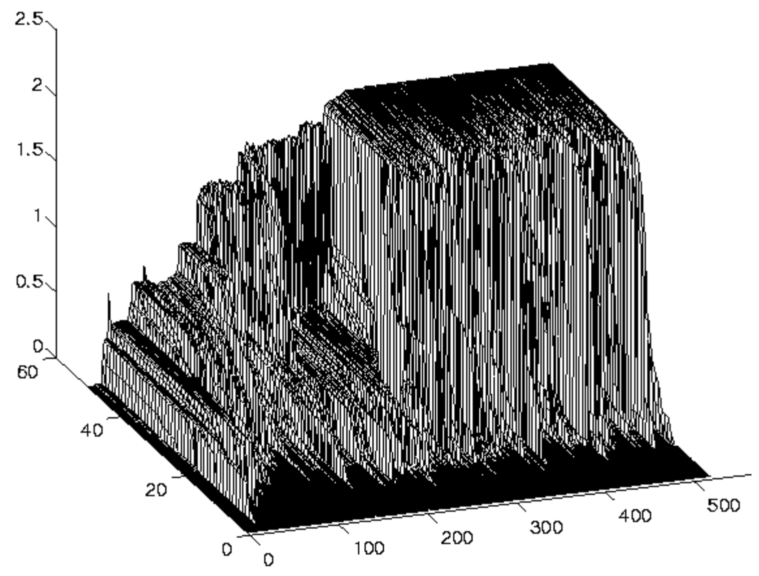

Figure 14: The fitness landscape, as sampled by the population. Over 500 snapshots are displayed from left to right (one every 10 generations). Within each such snapshot the 50 members of the population are ranked according to fitness, which is plotted on the vertical scale. After the initial stages a narrow ridge leads up to a flat plateau which starts around snapshot 266, which is generation 2660.

GA had not used the strategy of elitism, this type of ridge might not have occurred; this speculation has not been checked by experiment.

9 Lessons for Evolutionary Algorithms

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 GA 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.

It is suggested in this paper that long-term incremental evolution of the design of many classes of complex systems — robot control systems and molecular design as well as hardware — will have some important characteristics in common with this example. Genetically converged populations will be used, as in SAGA, which means that mutation rates must be carefully set to balance against selective pressure. Account must be taken of the proportion of 'junk DNA' when setting these mutation rates. Such redundant or junk DNA can under some circumstances be useful in allowing widely percolating neutral networks through genotype space, making entrapment on local optima highly unlikely.

Conventional GAs and GP, work on different principles, and are appropriate for a different set of problems; where an individual parameter set has to be optimised once and for all. There is a difference between such optimisation problems, and long-term incremental evolution.

10 Commercial Prospects

The incremental nature of evolution, both natural and artificial, means that any new adaptation builds on the inheritance accumulated from all its ancestors, without which it would have been inconceivable — literally as well as metaphorically. Such an inheritance can only be achieved through enormous numbers of trials, typically expensive. To achieve a complex system capable of coping with (in order of difficulty) tasks $T_1 \rightarrow T_8$ will have cost a lot; to go further from $T_8 \rightarrow T_9$ will cost significantly less than starting from the beginning and achieving all the capabilities $T_1 \rightarrow T_9$. This is equally true in the world of, for example, aircraft design, where nowadays no aircraft manufacturer could start up without buying in expertise derived from other's design and manufacturing experience.

Accumulated experience in artificial evolution can be expressed, given some specific genotype-to-phenotype mapping, simply in the string of numbers that form the genotype. This provides a basis for, and also limits and constrains, future pathways of evolution. The value of such a genotype can easily be protected by copyright or patent laws, and licensed for use by others. Future progeny can readily be identified as related by use of string-matching algorithms, so that pirating of copies can be traced. It seems likely that in the next century much design work for complex systems — not just robot control systems, but aircraft design, computer chip design, etc. — will start to be done by artificial evolution. The commercial prospects for any firm that achieves a head start in some field, and then licenses its genotypes for use by others with a licence fee payable on all progeny, could be phenomenal. Such a licence fee could be insignificant on any one unit; as evolution continues it could have additional small fees added for the benefit of those who have added value; long-term returns on all future progeny could nevertheless be enormous. As other firms build on and add value, for their own benefit, to the original genotypes, the commercial position of those genotypes relative to any competition becomes further strengthened — the founder effect translated directly to the commercial world.

References

[1] Hideki Asoh and Heinz 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] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, Reading MA, 1989.

[3] 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.

[4] I. Harvey, P. Husbands, D. Cliff, A. Thompson, and N. Jakobi. Evolutionary robotics: the Sussex approach. *Robotics and Autonomous Systems*, 1996 In Press.

[5] 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.

[6] 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: Proceedings of the Third International Conference on Simulation of Adaptive Behaviour (SAB94)*, pages 392–401. MIT Press/Bradford Books, Cambridge MA, 1994.

[7] I. Harvey and A. Thompson. Through the labyrinth evolution finds a way: A silicon ridge. In T. Higuchi, editor, *Proc. of The First International Conference on Evolvable Systems: From Biology to Hardware (ICES96)*. Springer-Verlag, 1996.

[8] Inman Harvey. The SAGA cross: the mechanics of crossover for variable-length genetic algorithms. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature, 2*, pages 269–278. North-Holland, 1992.

[9] Inman Harvey. Species adaptation genetic algorithms: The 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.

[10] 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.

[11] N. Jakobi. Half-baked, ad-hoc and noisy: Minimal simulations for evolutionary robotics. In *Submitted*, 1997.

[12] N. Jakobi, P. Husbands, and I. Harvey. Noise and the reality gap: The use of simulation in evolutionary robotics. 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*. Springer Verlag, 1995.

[13] J. R. Koza. *Genetic Programming*. MIT Press/Bradford Books, Cambridge MA, 1992.

[14] A. Thompson. An evolved circuit, intrinsic in silicon, entwined with physics. In T. Higuchi, editor, *Proc. of The First International Conference on Evolvable Systems: From Biology to Hardware (ICES96)*. Springer-Verlag, 1996.