This is a preprint of an article published in Complexity, Vol. 7, Issue 2, pp. 19-33, Wiley Periodicals, 2001

How neutral networks influence evolvability

Marc Ebner[1], Mark Shackleton[2] and Rob Shipman[2]

[1] Universität Würzburg, Lehrstuhl für Informatik II, Am Hubland, 97074 Würzburg, Germany
ebner@informatik.uni-wuerzburg.de

[2] BT Labs at Adastral Park, Admin 2-5, Martlesham Heath, Ipswich IP5 3RE, UK
{mark.shackleton,rob.shipman}@bt.com

Abstract

Evolutionary algorithms apply the process of variation, reproduction and selection to look for an individual capable of solving the task at hand. In order to improve the evolvability of a population we propose to copy important characteristics of nature's search space. Desired characteristics for a genotype-phenotype mapping are described and several highly redundant genotype-phenotype mappings are analyzed in the context of a population based search. We show that evolvability, defined as the ability of random variations to sometimes produce improvement, is influenced by the existence of neutral networks in genotype space. Redundant mappings allow the population to spread along the network of neutral mutations and the population is quickly able to recover after a change has occurred. The extent of the neutral networks affects the interconnectivity of the search space and thereby affects evolvability.

Neutral evolution, neutral networks, fitness landscapes, redundant mappings, changing environment, adaptation.

1 Introduction

Natural evolution differs in many respects from the way evolutionary algorithms are usually used in computer science. According to the neutral theory of evolution, a considerable fraction of all mutations are neutral and only a small fraction of non-neutral mutations are actually beneficial [22]. Some neutral mutations result from a redundancy in the genetic code but most redundancy arises in the development of the phenotype from the genotype. However, most evolutionary algorithms use a fixed one-to-one mapping between genotype and phenotype space. Each genotype corresponds to exactly one phenotype. In this article we describe the impact of redundant mappings on the search space. If an appropriate redundant mapping is used, more phenotypes become accessible which can help to make the search easier. A redundant mapping induces neutral networks, which are sets of genotypes that represent the same phenotype and are connected by single point mutations [17, 16, 43].

1.1 The impact of neutral nets on evolution

Suppose we are using a one-to-one mapping between genotype and phenotype space and that point mutations are the only way that variations can be introduced into the population. In this case the number of phenotypes reachable from any given genotype is exactly $(A-1)L$ [20] where A is the number of alleles per loci and L is the length of the genotype. If this genotype happens to be a local optimum, i.e. none of the neighbors are better than the current one, then we are stuck and

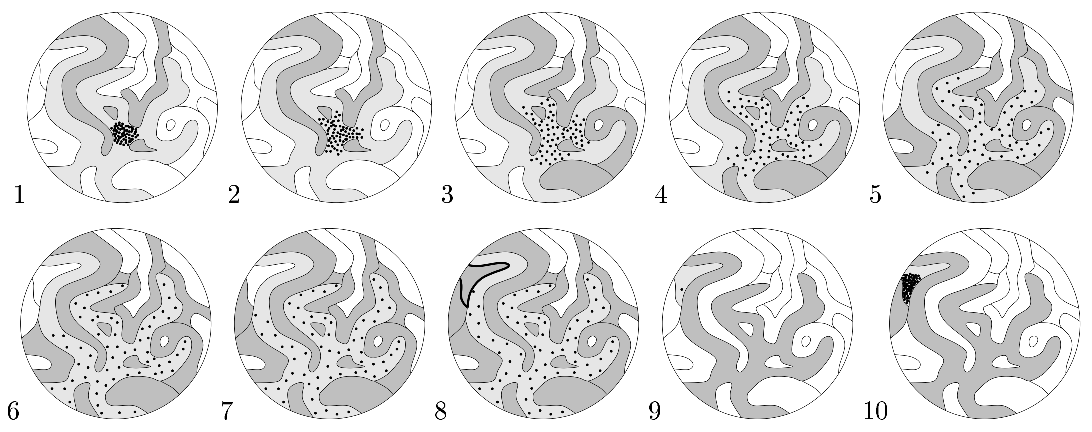


Figure 1: Initially all individuals are located in a small area of genotype space. Individuals are represented as dots. Lines in this space denote borders between different phenotypes. It is assumed that only a small neighborhood can be reached by using a point mutation and that the light gray area corresponds to the fittest phenotype. Individuals which fall outside the light gray area quickly die out. The dark gray areas correspond to phenotypes which can be reached by a single point mutation from at least one member of the current population. Over time the population spreads along the neutral network and more and more adjacent phenotypes become reachable via point mutations. If the fitness landscape changes, only the one mutant neighbors of the genotype closest to the now fittest phenotype survive, and the process starts anew. This happens at the 8th time step of the above sequence, where the area enclosed by the thick line is to become the fittest phenotype due to a change of the fitness landscape. In the absence of neural nets no individual would have been within one (or a few) mutations to the new environment. Adaptation to the new environment, consequently, would be much slower or even nonexistent.

no further improvement is possible. However, if a redundant mapping is used, more phenotypes may become accessible via neutral networks because it is possible to drift along neutral mutations to new regions in the search space. The number of phenotypes reachable from this neutral network may be much greater than the number of neighbors given a non-redundant mapping.

The benefit of this increase in reachable phenotypes is that a search algorithm is more likely to be able to reach a superior phenotype (without having to traverse through inferior phenotypes) and thus it has an increased chance of avoiding becoming trapped in a local optima. The operation of a population based algorithm on a landscape with neutrality has additional interesting properties. For example, consider the case where the algorithm has found an optimal solution and suppose that the population is spread out through the neutral network of genotypes that map to this phenotype. If then the fitness landscape should change, and some new phenotype becomes optimal, there is a chance that some member of the population will already be quite close to a genotype that maps to this new best phenotype. The important thing to notice is that an evolutionary algorithm does not only operate on a single individual. If a population of individuals is used, the individuals are able to spread along paths of neutral mutations. How this works is illustrated in Figure 1. Therefore the number of phenotypes reachable via point mutations can be much larger if an appropriate redundant mapping is used and the population is sufficiently spread out along a neutral network. But what type of redundant mapping should be used? To answer this question one can draw inspiration from nature's search space.

1.2 Desired characteristics of the search space

Schuster [34] has analyzed the shape space of RNA secondary structures. According to Schuster there are few shapes that are common and many shapes are rare. Sequences that code for all common shapes are located in the vicinity of any randomly selected sequence. In addition, different sequences that code for a specific shape are randomly distributed in sequence space. Long paths of neutral mutations lead to sequences which code for identical shapes. Huynen [16] and Huynen et al. [17] showed that neutral evolution plays a role in adaptation in the context of fitness landscapes based on the mapping between RNA sequence and RNA secondary structure. Ebner [6] and Shipman [36] have argued that similar characteristics might also be useful for an artificial search algorithm. A number of studies were carried out to analyze the benefits of a highly redundant genotype-phenotype mappings which show similar characteristics to nature's search space. Shipman et al. [37] analyzed the search space created by several redundant genotype-phenotype mappings with different amounts of redundancy. Shackleton et al. [35] introduced the concept of fitness to the redundant mappings and performed fitness adaptive walks for a single point moving through genotype space. Ebner et al. [7] have shown that highly redundant mappings increase evolvability, defined as the ability of random variations to sometimes produce improvement, in a population of co-evolving species. Bullock [5] analyzed mutation bias of redundant mappings.

Other research exploring the effects of redundancy include the work of Barnett [4], who introduced redundancy into Kauffman's NK fitness landscape [20] and analyzed population dynamics on this static fitness landscapes. Newman and Engelhardt [33] also presented a variant of the NK landscape with neutrality. In both cases, the amount of redundancy inherent in the landscape may be tuned by varying a parameter. Levenick [29] has looked at the advantages of having introns for a dynamic environment. Julstrom [18] established that redundancy is beneficial in looking for a solution to the problem of partitioning $3n$ points into 3-cycles of minimum total length. Recently, Yu and Miller [56] investigated the benefits of neutral evolution for the synthesis of digital circuits.

Banzhaf [2] studied a simple redundant map in the context of linear computer programs and Keller and Banzhaf [21] evolved a genotype-phenotype map for computer programs. The effects of a simple redundant mapping on the search space were analyzed theoretically by Kargupta [19]. The role of development in genetic algorithms has been analyzed by Hart et al. [14]. Other researchers have experimented with developmental mappings where the genotype specifies a grammar from which the phenotype develops [12, 13, 25]. In the genetic programming paradigm [23, 24] introns usually occur at the end of an experiment after a local optimum has been reached [3]. Individuals can increase the probability that their offspring will have the same fitness as their parent by increasing the number of introns. This process is usually referred to as bloat in the context of genetic programming [27, 28]. Code growth is studied in detail by Smith and Harries [40]. That introns may actually be useful to the evolution of computer programs was first suggested by Angeline [1]. The benefits of introns for program induction was investigated by Wineberg and Oppacher [51].

In this article, we perform a detailed analysis of the structure and extent of neutral networks inherent in certain types of genotype-phenotype mappings. Not all mappings have the right type of redundancy. Simple types of redundancy without extensive and highly intertwined neutral networks simply slow the rate of finding adaptive mutations. An example of this type of redundancy includes a genotype with non-coding bits. A change only occurs if one of the expressed bits is mutated. The mappings presented here have a different type of redundancy. They contain highly intertwined neutral networks which provide some of the benefits of nature's search space for an evolutionary algorithm. The article consists of four separate sections. First we describe the redundant mappings. After we have described the mappings, we perform a detailed statistical analysis on the structure of the neutral networks. Next, we perform a population based search on a static fitness landscape. And finally, we investigate the impact of neutral networks on a dynamic fitness landscape.

2 Mappings

We looked at several different genotype-phenotype mappings. Two of the mappings which we investigated, produced particularly interesting results: a mapping based on a cellular automaton, and a mapping based on a random boolean network. These mappings possess extensive and

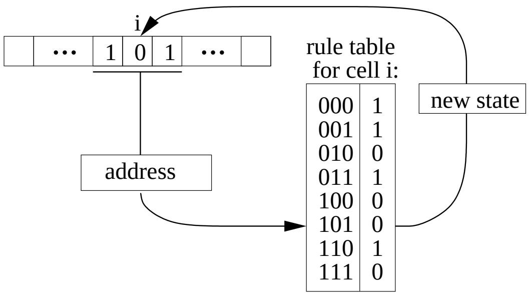


Figure 2: A linear non-uniform cellular automaton. The next state of the cell i is determined by the state of cell i as well as the state of its neighboring cells $i-1$ and $i+1$ (with circular boundary conditions). The state of these three cells specify an address in the rule table of cell i which determines the state of cell i at the next time step.

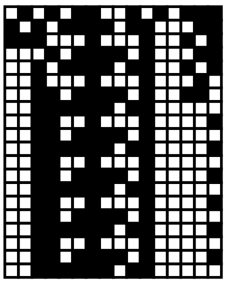


Figure 3: A sample run of a cellular automaton with 16 cells for 20 steps. The initial state of the automaton was a fixed random value.

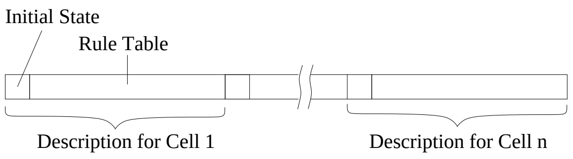


Figure 4: Genotype structure for the cellular automaton mapping.

highly intertwined neutral networks . A standard binary mapping is used to provide a baseline. The cellular automaton mapping and the random boolean network mapping have been analyzed previously by Shipman et al. [37] and Shackleton et al. [35]. The mappings differ in the amount of redundancy used. In this case, the measure of redundancy is the ratio of the number of genotypes to the number of phenotypes. The binary mapping is a one-to-one mapping. The cellular automaton mapping has a redundancy of $2^{64}:1$ (for a phenotype space of 8 bits), and the random boolean network mapping has a redundancy of $2^{136}:1$ (for a phenotype space of 8 bits).

2.1 Binary mapping

For the binary mapping, genotype space and phenotype space are equivalent. The same number of bits are used for the genotype as well as for the phenotype. There exists a one-to-one correspondence between genotype and phenotype. The phenotype number is obtained by interpreting the genotype as a binary number.

2.2 Cellular automaton mapping

A linear cellular automaton consists of an array of cells where each cell has two possible states: on or off. The state of a cell at the next point in time is determined by the state of cell at the current time step and by the state of its neighboring cells [52, 53]. For a non-uniform cellular automaton, each cell has its unique rule table [39]. This rule table defines the next state of the cell given the current state value of the cell and its neighbors. We have used a neighborhood of 2. Therefore the rule table of each cell has 2^3 entries. How the cellular automaton works is described in Figure 2. A sample run of a cellular automaton with 16 cells is shown in Figure 3.

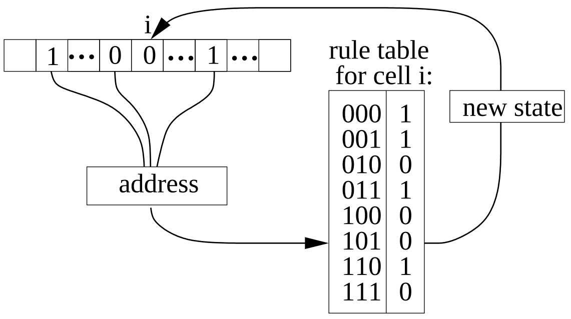


Figure 5: A random boolean network. The next state of cell i is determined by the state of the cells it is connected to. The state of these cells specify an address in the rule table of cell i which determines the state of cell i at the next time step. For our experiments all cells are connected to three other cells.

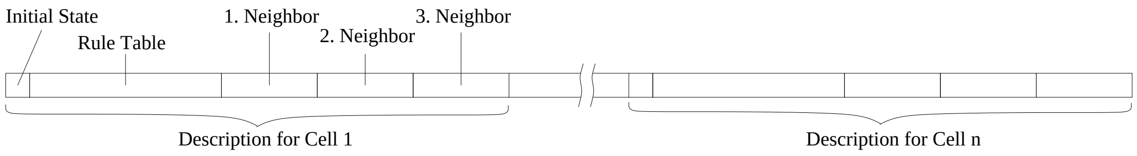


Figure 6: Genotype structure for the random boolean network mapping.

A cellular automaton can be considered as a model of ontogenetic development [26]. For the cellular automaton mapping, the genotype specifies the rule tables of the automaton and the initial state of each cell. A variant of this mapping was presented for the first time in Shackleton et al. [35]. The phenotype is determined by running the cellular automaton for a fixed number of iterations (20 in our experiments). Let n_p be the number phenotypes then we need an automaton with $n = log_2 n_p$ states. After the automaton is run for 20 iterations we interpret the resulting state as a binary number. This number is our phenotype. Because the rule table of the automaton and the initial state of each cell is stored in the genotype we need $n(8+1)$ bits for the genotype. The structure of the genotype for the cellular automaton mapping is shown in Figure 4. If we need to determine the phenotype for a given genotype we first read off the initial states of the cells and initialize the cellular automaton with these states. Next we decode the rule tables for the cells of the automaton and run the automaton for 20 iterations. The resulting state interpreted as a binary number is the number of our phenotype.

2.3 Random boolean network mapping

A random boolean networks consists of a fixed number of cells [20]. Each cell has two possible states: on or off. The state of a cell at the next point in time is determined by the state of the cells it is connected to. Each cell has its own rule table which defines the next state of the cell given the current state values of the cells it is connected to. For our experiments we have used a connectivity of 3. Therefore the rule table of each cell has 2^3 entries. How a random boolean network works is shown in Figure 5.

For the mapping based on a random boolean network, the genotype specifies the initial state, its wiring and all of the rule tables. The mapping was introduced by Shipman et al. [37]. The phenotype is determined by running the random boolean network for a fixed number of iterations (20 in our experiments). After that, the resulting state of the network is interpreted as the number of the phenotype. Let n be the number of cells of the random boolean network. Each cell has n_c cells to which it is connected. Then we need $n(1 + n_c \log_2(n) + 2^{n_c})$ bits to describe the random

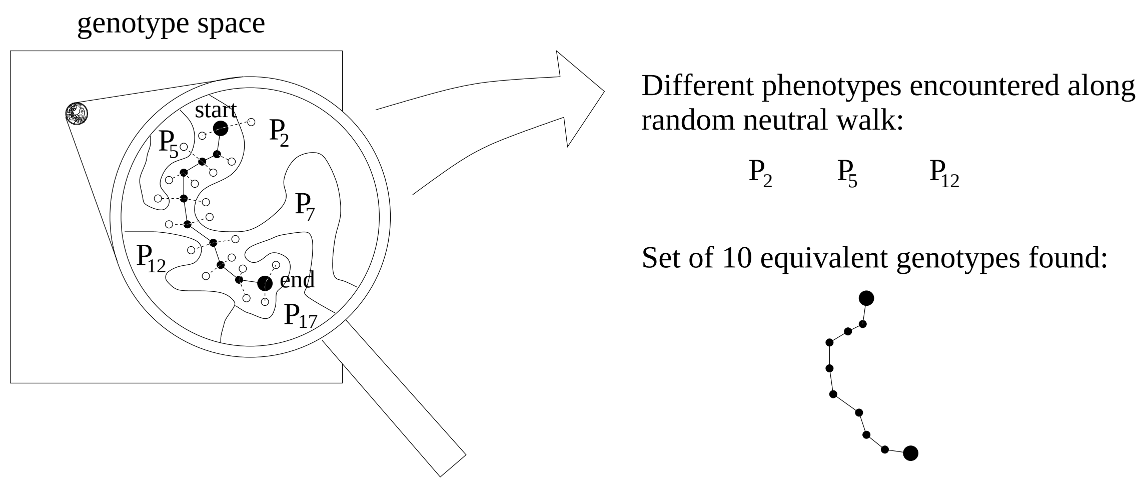

genotype space
start
end
P_2
P_5
P_7
P_{12}
P_{17}
Different phenotypes encountered along random neutral walk:
P_2 P_5 P_{12}
Set of 10 equivalent genotypes found:

Figure 7: A Random neutral walk in genotype space. Starting from a given genotype a number of neutral mutations are performed which lead us through genotype space. During this random neutral walk we encounter a number of genotypes which are reachable using point mutations which map to different phenotypes. As a result of the random neutral walk we obtain a list of reachable phenotypes and a list of equivalent genotypes all mapping to the same phenotype.

boolean network. For each cell we need one bit to specify the initial state, $n_c \log_2(n)$ bits to specify to which cells it is connected to, and 2^{n_c} bits for the rule table. The structure of the genotype for the random boolean network mapping is shown in Figure 6.

3 Statistical analysis

To evaluate the different mappings we performed a statistical analysis. After we have performed this analysis we will be able to make some predictions on the characteristics of the mappings. The analysis will give us detailed information on the extent and the structure of the neutral networks. We will then look at the mappings in an evolutionary setting in the next section. A set of statistics has been developed to evaluate the characteristics of the different mappings. In order to be able to completely analyze the genotype-phenotype mappings we limited the number of phenotypes to 256 as this allows us to compute statistics over all phenotypes. In this case, the cellular automaton mappings has a redundancy of $2^{64} : 1$, and the random boolean network mapping has a redundancy of $2^{136} : 1$.

3.1 A random neutral walk through genotype space

In order to analyze the neutral network we perform a random neutral walk through genotype space (Figure 7). This is our main tool to investigate the connectivity and extent of the neutral networks. A random neutral walk starts from a particular genotype and determines all one mutant neighbors. The walk proceeds by randomly selecting one of these mutants which maps to the same phenotype. In case no neutral neighbor is found, as could happen with a standard one-to-one mapping, the walk remains at the current position in genotype space. This process continues for a specified number of steps. As a result we get a list of reachable phenotypes via point mutations and a list of genotypes that have equivalent phenotypes.

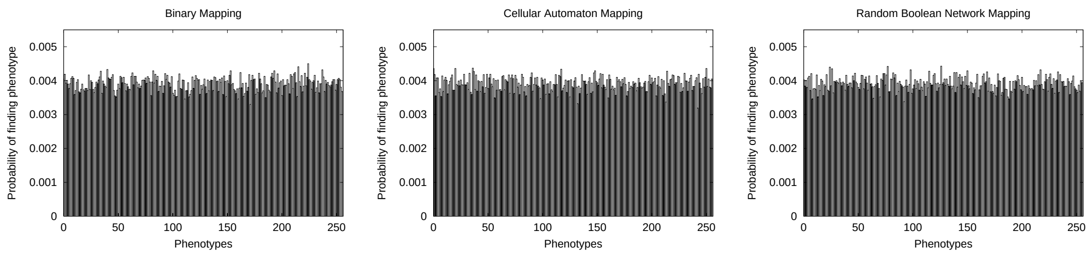


Figure 8: Probability of locating any of the phenotypes in the solution space using random search.

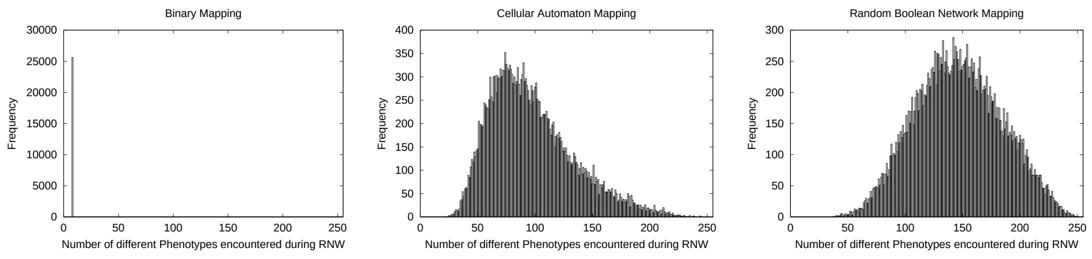


Figure 9: Number of phenotypes reachable across all random neutral walks.

3.2 Probability of locating a phenotype using random search

Depending on the type of mapping used, different phenotypes may cover different amounts of genotype space. The first statistic determines the amount of genotype space covered by the different phenotypes. We randomly created 100000 different genotypes and determined the corresponding phenotype. We then tabulate how many times each phenotype was found during this random search in genotype space. The results are shown in Figure 8. For all mappings the probability distribution of locating a given phenotype using random search is almost flat. Thus no phenotype covers an excessive amount of genotype space. If this were the case, then any search would be highly biased towards this phenotype.

3.3 Reachability of phenotypes

The reachability of a given phenotype from a random phenotype is estimated as follows. We perform a series of random neutral walks (100 for every phenotype) and tabulate how many phenotypes were encountered for every walk. The histograms for this experiment are shown in Figure 9. For the binary mapping 8 phenotypes can be reached. For the cellular automaton mapping approximately 100 other phenotypes are reachable during a random neutral walk. For the random boolean network mapping, most walks encounter about 150 reachable phenotypes and almost none have less than 50 reachable phenotypes. If we look at the frequency of encountering specific phenotypes shown in Figure 10 we again get a flat distribution. No preference is given to any phenotype. Each phenotype is encountered 8 times for the binary mapping, on average 100 times for the cellular automaton mapping and 150 times on average for the random boolean network mapping (assuming a walk length of 100).

3.4 Innovation rate

The number of new phenotypes encountered as a function of walk length, is calculated as follows. For every phenotype we select a random genotype which maps to this phenotype. Next we perform a random neutral walk and save the cumulative number of phenotypes reachable along the walk.

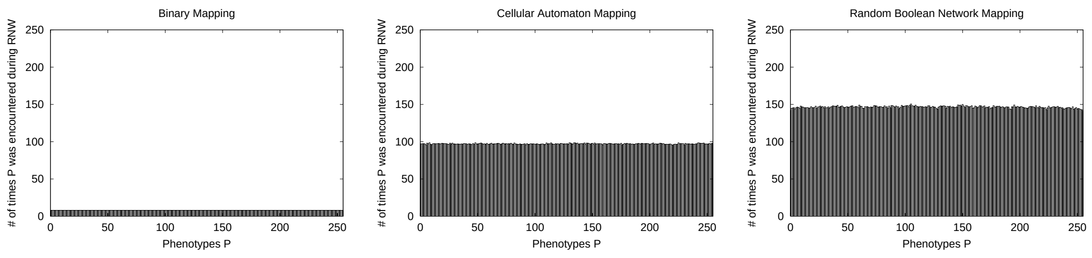


Figure 10: Reachability distribution for individual phenotypes across all random neutral walks.

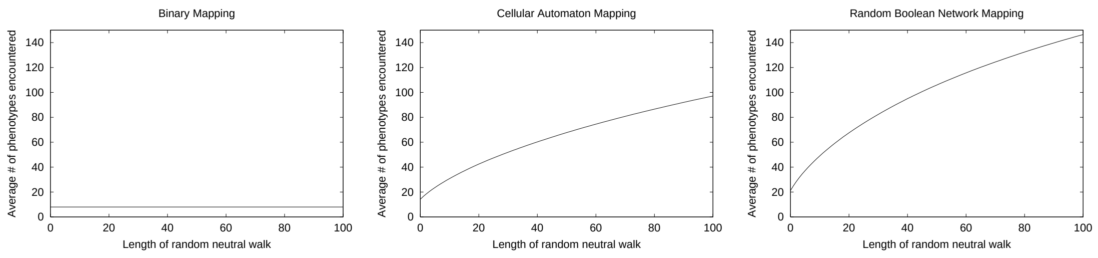


Figure 11: Number of new phenotypes encountered as a function of walk length.

We performed 100 independent walks each of length 100 for each phenotype and averaged the result over all walks. With a binary mapping no additional phenotypes can be reached because there are no neutral mutations for the binary mapping. For the cellular automaton mapping as well as for the random boolean network mapping a large number of other phenotypes are accessible right from the start. As the random walk proceeds more phenotypes become accessible. It is interesting to note that, in both cases, new phenotypes are continually discovered. The slope of the graph can be considered to be the innovation rate [16]. That is, the rate of discovering new phenotypes when taking a step along the neutral network. Whereas for the binary mapping we get an innovation rate of zero, for the two redundant genotype-phenotype mappings we get an innovation rate of 2.24 and 3.79 respectively at the beginning of the walk. The innovation rate should be a good indicator of the evolvability of a population evolving in the context of a particular mapping.

3.5 Connectivity between phenotypes

We now look at the connectivity between phenotypes. This connectivity is calculated as follows. For every phenotype we select 100 random genotypes which map to this phenotype. Next we perform random neutral walks starting from each of these genotypes and mark the phenotypes reachable for this phenotype. The connectivity matrix for a walk length of 100 steps is shown in Figure 12. For the binary mapping, connectivity is very low. Each phenotype connects to eight others via a one-point mutation and one can clearly see an underlying structure in the connectivity matrix. For the cellular automaton mapping and the random boolean network mapping one is able to reach any other phenotype due to the extensive neutral networks of this mapping. In order to better visualize the structure we now count the number of times a phenotype was accessed. Figure 13 shows the frequency distribution of the connectivity matrix with a logarithmic scale. This method was suggested by Bullock [5] in an effort to better visualize the structure of the matrix. That is, whereas Figure 12 shows that in principle each phenotype can reach any other phenotype, Figure 13 shows the probability of encountering a particular phenotype from a given phenotype along the random walk.

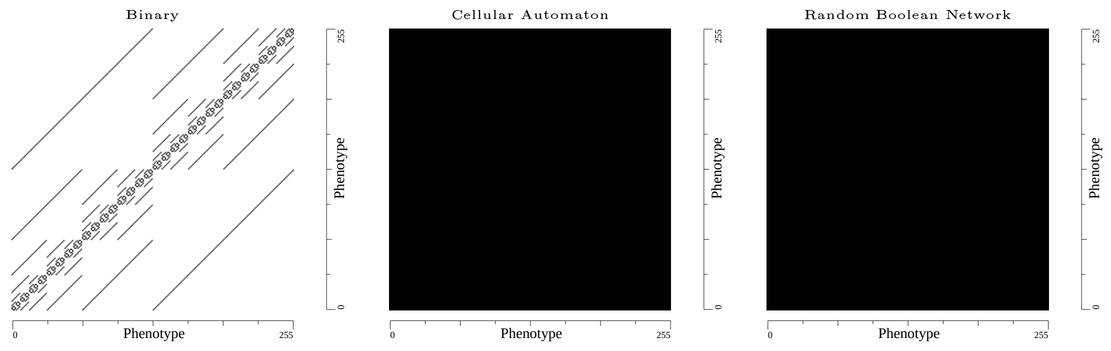


Figure 12: Connectivity between phenotypes

.

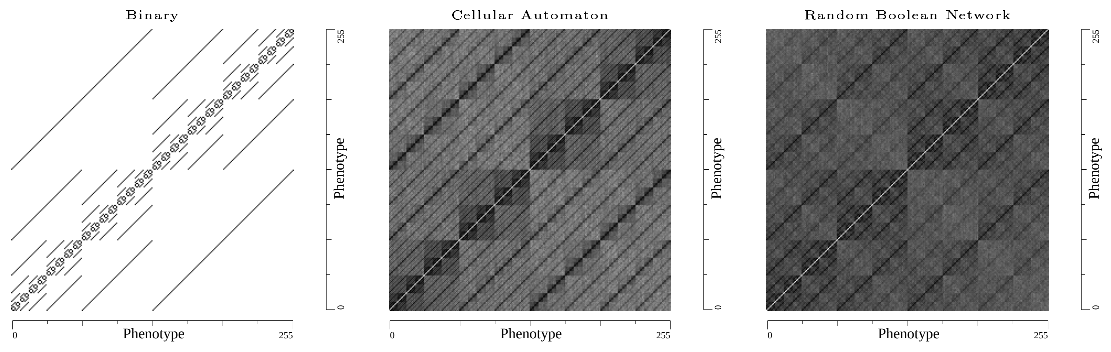


Figure 13: Probability of reaching another phenotype from a given phenotype (log scaled).

3.6 Extent of the neutral networks

We now look at the extent of the neutral networks. Again we perform a series of random neutral walks (100 walks each of length 100) and measure the maximum Hamming distance between the original genotype and the phenotypically equivalent genotypes found during the random neutral walk. Thus, we measure how far we moved through genotype space. For the cellular automaton mapping 32 bits had changed on average after 100 steps and for the random boolean network mapping 49 bits had changed. It is easier to move through genotype space on the random boolean network mapping. In case of the cellular automaton mapping 44% and for the random boolean network mapping 58% of all mutations are neutral. For the cellular automaton mapping each genotype has on average access to 14 phenotypes and for the random boolean network mapping each genotype has access to 21 phenotypes via one point mutations. Via neutral mutations eventually all phenotypes become accessible as was shown in Figure 12.

3.7 Prediction from the statistical analysis

We would expect that for a particularly good mapping the probability of reaching a given phenotype from a random phenotype is high. In addition, we would like to have a high innovation rate, that is the graph showing the number of phenotypes reachable as a function of walk length should have a high slope at the origin and maintain a high slope until eventually all phenotypes have become accessible. Results are summarized in Table 14. The statistics are especially encouraging for the random boolean network mapping. This mapping has a very high accessibility of phenotypes reachable via random neutral walks and a high innovation rate. Whereas the binary

Mapping	Reachable Phenotypes	Innovation Rate	% of neutral mutations	Connectivity Matrix	Extent of the networks
Binary	8	0	0	almost empty	no networks
CA	100	2.24	44%	high connectivity	large and intertwined
RBN	150	3.79	58%	high connectivity	extensive and highly intertwined

Figure 14: Summary of the statistics for the different types of genotype-phenotype mappings. The results are shown for a single individual moving through genotype space with a walk length of 100.

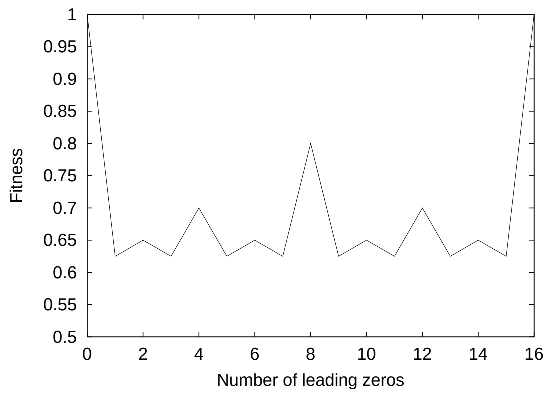


Figure 15: Cut through the "hierarchical if and only if" landscape.

mapping exhibits no neutrality at all, the random boolean network mapping contains extensive neutral networks which are highly intertwined. Thus, from the statistical analysis we assume that the cellular automaton mapping would perform better than the binary mapping and that the random boolean network mapping would perform best.

4 Experiments on a static landscape

The above statistical analysis suggests that the mapping based on the random boolean network is a particularly good mapping. We now experiment with a population of individuals adapting to a static fitness landscape. We will see that evolvability (defined as the ability of random variations to sometimes produce improvement [44]) of a population is dependent on the genotype-phenotype mapping used. If our search space has similar characteristics as nature's search space as described above, the individuals should have little problem discovering phenotypes with better fitness values. We will see that this is indeed the case.

4.1 A hierarchical fitness function

For our experiments on a static landscape we have selected the "hierarchical if and only if" fitness function. The hierarchical if and only if function was introduced by Watson et al. [46, 47, 48, 49]. It was especially constructed to show what kind of problems a genetic algorithm with crossover is suitable for. This function has a very large number of local optima and a mutation-only algorithm cannot be guaranteed to succeed in time less than exponential in the number of bits used for the problem [45]. A cut though the hierarchical if and only if fitness landscape is shown in Figure 15.

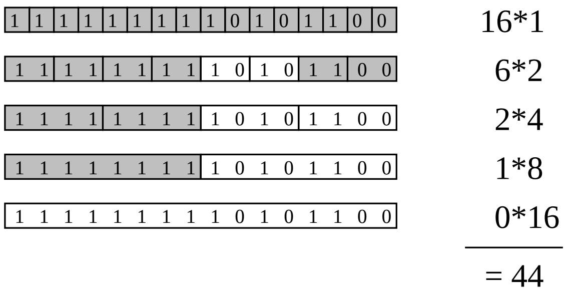


Figure 16: Explanation of fitness assignment using the hierarchical if and only if function.

The hierarchical if and only if function is defined as follows:

$$f(b_1, b_2, ..., b_n) = \begin{cases} 1 & \text{if } n = 1 \\ n + f(b_1, ..., b_{n/2}) + f(b_{n/2+1}, ..., b_n) & \text{if } n > 1 \text{ and } (\forall i : b_i = 0 \text{ or } \forall i : b_i = 1) \\ f(b_1, ..., b_{n/2}) + f(b_{n/2+1}, ..., b_n) & \text{otherwise} \end{cases}$$

The function consists of a hierarchy of building blocks. On the first level each bit receives 1 towards fitness irrespective if the bit is one or zero. On the next level, we create groups of two and award 2 towards fitness for every group which consists of bits with equal values. On the next higher level we create groups of 4 bits and award 4 towards fitness for every group in which all bits are equivalent. This process continues until we have one large group which contains all the bits (Figure 16). Maximum fitness is reached if all bits are zero or if all bits are one. Thus, different building blocks need to be combined to solve a sub-problem. The solutions to this sub-problem again need to be combined to solve the problem on the next step of the hierarchy. A genetic algorithm using mutation and crossover is especially suited to solve this problem provided that diversity is maintained [46].

To each phenotype we assigned a fitness value using the hierarchical if and only if function. A phenotype space of 16 bits was used. Thus, the cellular automaton mapping has a redundancy of $2^{128} : 1$. In this case the genotype consists of 144 bits. The random boolean network mapping has a redundancy of $2^{320} : 1$ and the genotype consists of 336 bits. For each genotype-phenotype mapping we ran the following experiment. A population of 100 individuals is randomly distributed over genotype space. Each individual performs an adaptive walk through genotype space. On each step we randomly mutate the individual. Next we determine the phenotype and the fitness value of this phenotype. If the mutated genotype has a higher or equivalent fitness value we keep this genotype as the current genotype for this individual. Thus, if no adaptive moves to a higher fitness value are possible, the individual nevertheless keeps moving through genotype space possibly encountering a point with higher fitness at a later point in time. The results of this experiment are shown in Figure 17. The graph on the left shows the maximum, average, and minimum fitness values of all individuals during the run. The graph on the right shows the fitness values of all 100 individuals. As can be seen from the graphs, the population on the binary mapping is unable to adapt to this landscape. It quickly becomes stuck in a local optimum. Evolution has come to a halt. However, the population using the cellular automaton mapping reaches a much higher fitness value. The random boolean network mapping performed similarly but not quite as well as the cellular automaton mapping. The hierarchical if and only if function was especially constructed to show which type of problems genetic algorithms are especially suited for. The above results show that the hierarchical if and only if problem with 16 bits can also be solved using a hill climber provided that a genotype-phenotype mapping with extensive neutral networks is used.

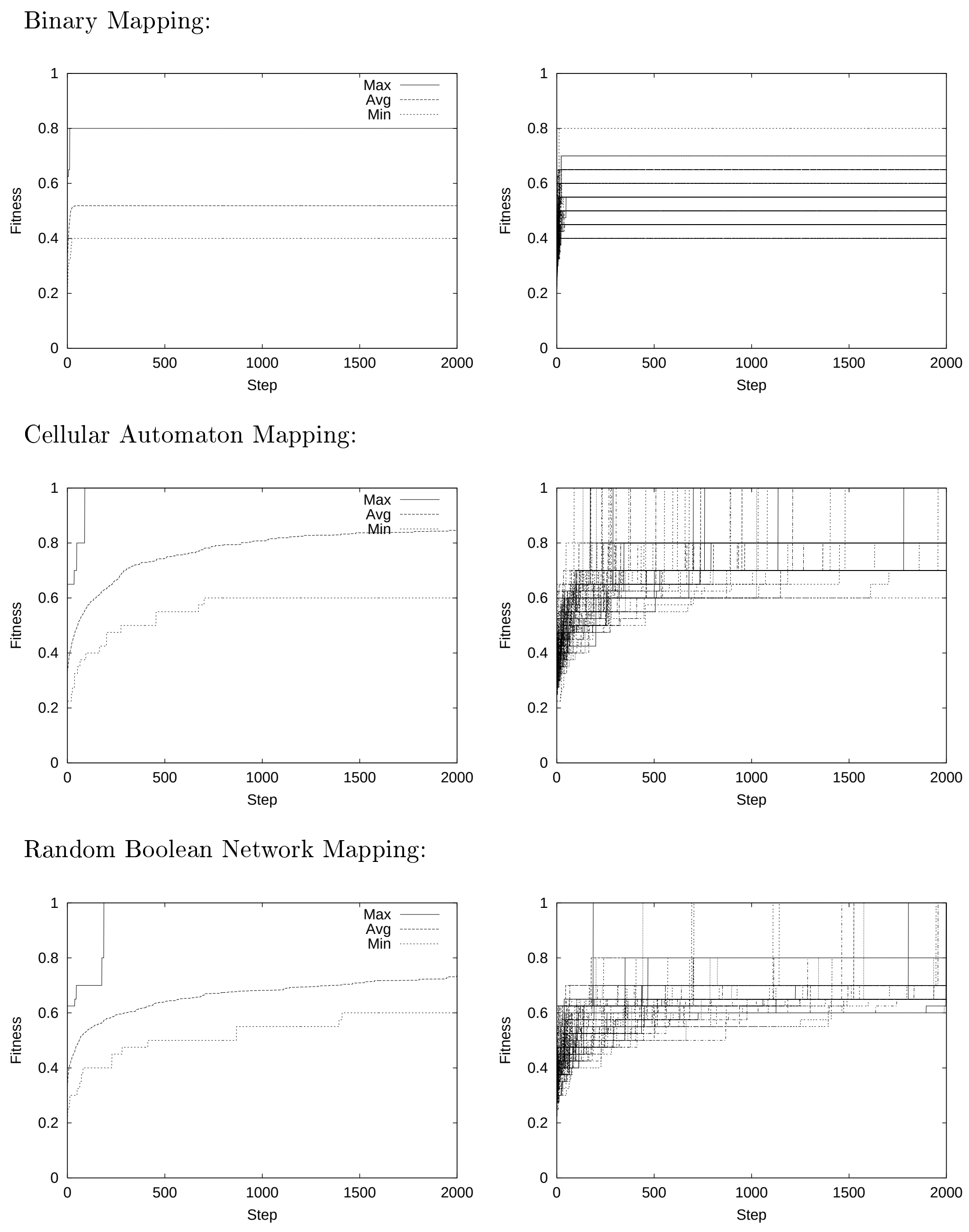


Figure 17: Adaptation of a population of individuals on the hierarchical if and only if function. Results are shown for three different genotype-phenotype mappings: binary mapping, cellular automaton mapping and random boolean network mapping. The graphs on the left show the maximum, average and minimum fitness of the population and the graphs on the right show the individual fitness curves.

4.2 What if the mapping between phenotype and fitness is not well behaved?

The hierarchical if and only if function is a particularly difficult function to solve using a hill climber. Let us now look at a much simpler problem: a Lorentz function (Figure 18). This unimodal function can be solved easily by following the gradient to the top of the hill. However,

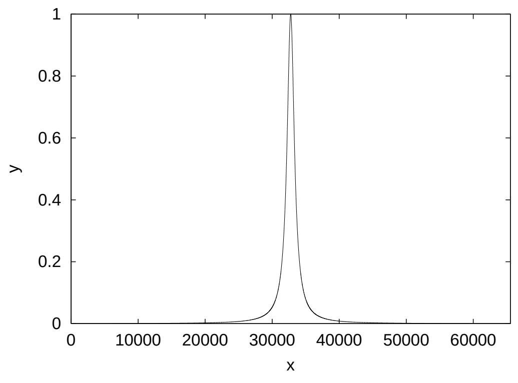


Figure 18: Unimodal fitness landscape.

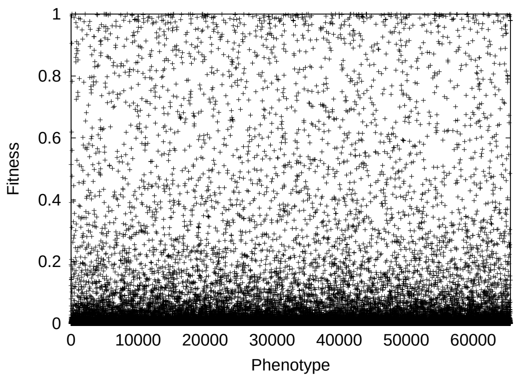


Figure 19: Shuffled Lorentz function.

this usually assumes that we have a well behaved coordinate system. What would happen if we had the same fitness landscape but random assignments between phenotypes and fitness values? This amounts to shuffling the fitness values of the landscape. Figure 19 shows what happens to the Lorentz function if we randomly shuffle the fitness values. This fitness landscape, like the hierarchical if and only if function, has many different local optima. Most phenotypes have a fitness value of zero and high fitness values are distributed all over the landscape. Note that in this case, we basically have a random landscape.

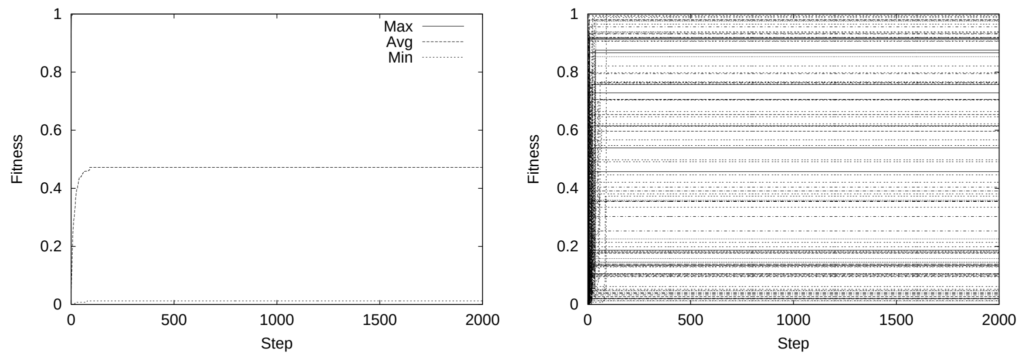
We now conduct the same experiment as for the hierarchical if and only if function but on the shuffled Lorentz function. The results of this experiment are shown in Figure 20. The random boolean network mapping performed best, followed by the cellular automaton mapping. The population on the binary mapping soon becomes stuck in a local optimum. Thus, we have seen that if the mapping between phenotypes and fitness values is unknown it pays to have a mapping with extensive neutral networks. The neutral networks create ridges between the local optima along which a hill climber can find its way to the top. The neutral networks provide additional accessibility between phenotypes and allows evolution to occur irrespective of the ruggedness of the original landscape.

5 Experiments on a dynamic landscape

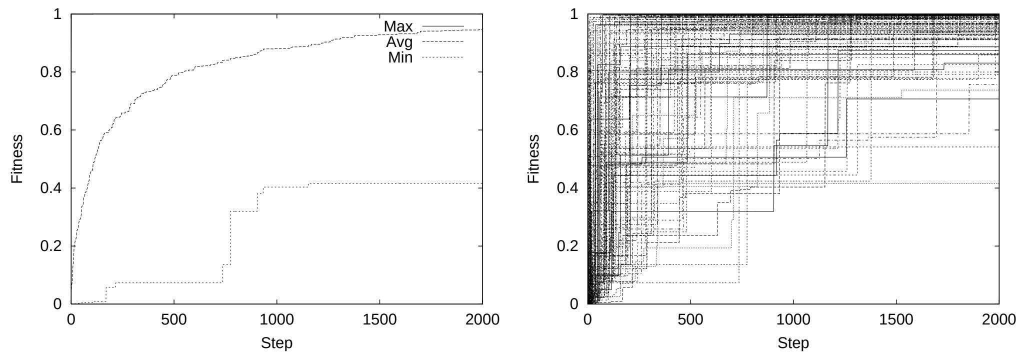
The above experiment has shown that neutral networks increase evolvability on a static landscape. We now look at how the population behaves on a dynamic landscape. Interest in dynamic fitness landscapes and tracking optima in a changing environment has gained considerable interest recently [8, 11, 29, 30, 32, 41, 42, 50]. With a non-redundant mapping, the population converges to the locally optimal genotype. If the environment changes, only the immediate neighbors are accessible using point mutations. The crossover operator is of no use for a converged population. The behavior of a population evolving with a redundant mapping is completely different. In principle, the population is able to spread along the neutral networks and thereby maintain diversity in case the environment should change. According to the no free lunch theorem, no algorithm will perform better than random search if one considers the space of all possible problems [54]. A redundant mapping ensures that this search is still possible even after the optimum has been found. Thus, a population evolving on a redundant mapping will always be able to perform this random search whereas a population on a non-redundant mapping will perform worse than random search after convergence has occurred.

We now look at how a population of individuals adapts to a changing environment. For this experiment we have used a standard genetic algorithm [15, 9, 31] and monitored maximum, average and minimum fitness of the population during the run. The mapping from phenotype to fitness was chosen at random as for the static case described above. Two different fitness landscapes were

Binary Mapping:



Cellular Automaton Mapping:



Random Boolean Network Mapping:

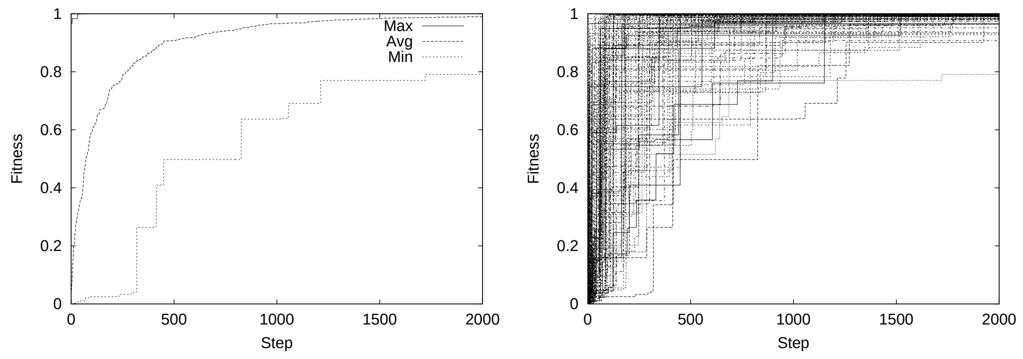


Figure 20: Adaptation of a population of individuals on the shuffled Lorentz fitness landscape. Results are shown for three different genotype-phenotype mappings: binary mapping, cellular automaton mapping and random boolean network mapping. The graphs on the left show the maximum, average and minimum fitness of the population and the graphs on the right show the individual fitness curves.

used to assign fitness values. Each fitness landscape is a shifted version of the Lorentz function. When the environment changes, the peak shifts from the left half to the right half of phenotype space. Note that we only change the fitness values. The structure of the neutral networks remains constant. The first landscape was used for generations 0 through 99 and the second landscape was used for generations 100 through 200 (Figure 21). This simulates the occurrence of a change

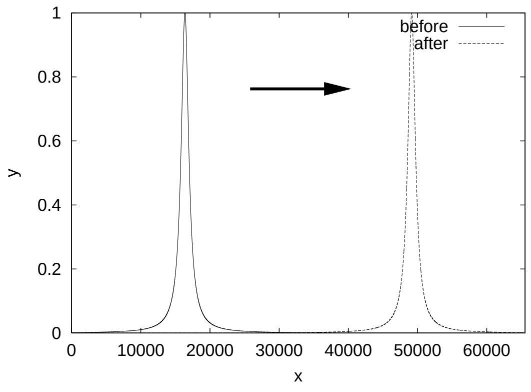


Figure 21: Changing environment. The optimum moves to a new position.

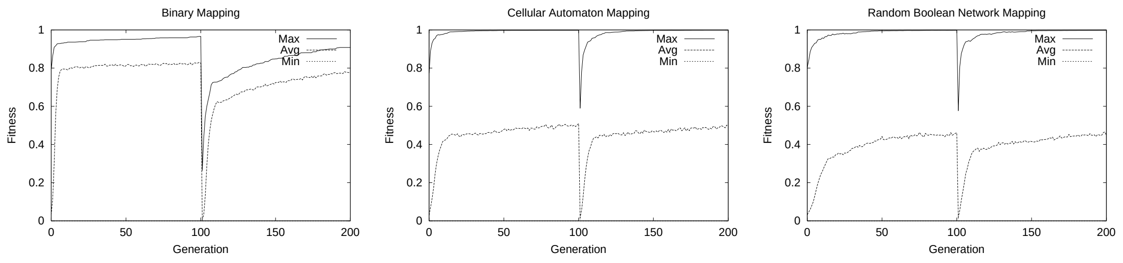


Figure 22: Performance of the population in a changing environment. Note that the minimum is so close to zero that it cannot be seen in the graph.

in the environment of the individuals. Each run was stopped after 200 generations. Results were averaged over 100 independent runs with different random seeds. For each run we used a population size of 100 individuals. Crossover rate was set to 0.5 and the mutation rate was set to 0.01 per locus. To select individuals we have used tournament selection with a tournament size of 8.

The results of this experiment are shown in Figure 22. The graphs show the maximum, average and minimum fitness of the population. As can be seen from the graphs, individuals evolving on the redundant mappings performed much better than the individuals evolving on the binary mapping. The maximum fitness of the population evolving on the redundant mapping did not drop as low compared to the population evolving on the binary mapping. The population evolving on the redundant mappings performed particularly well after the environment changed. Maximum fitness quickly recovered to the global optimum. The population on the binary mapping does not reach the original level during the allotted time. Again, the results are in accordance with the statistical analysis which was performed above. Use of redundant mappings have an additional advantage. If the mutation rate is high, the population is able to maintain the optimum once it is found. For high mutation rates, the population on a non-redundant mapping forms a cloud around the optimum [20, 55]. The height of the cloud is determined by the mutation rate which pulls the population away from the optimum and by selection which pulls the population towards the optimum. If a large number of mutations are neutral then the population may remain on the optimum even if the mutation rate is high.

A measure from population genetics [38] was used to measure the diversity of the population during the run. Diversity is calculated as the average probability that two randomly selected individuals differ at a randomly chosen locus. The results, again averaged over 100 runs, are shown in Figure 23. Selection pressure quickly reduces the diversity of the population for the

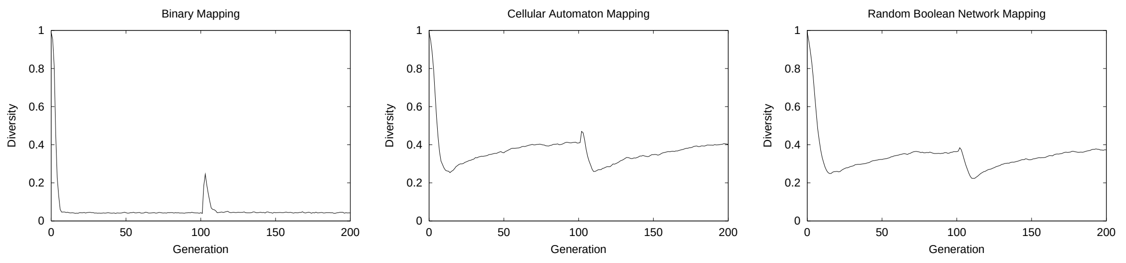


Figure 23: Diversity of the population.

binary mapping. Thus the population is converged in genotype space and further search becomes difficult. In comparison, the population evolving on the redundant mappings are able to maintain a much higher level of diversity. In all three cases, diversity briefly rises immediately after the environment has changed and then drops dramatically. In case of the redundant mapping one can clearly see that diversity increases and finally reaches the original level as the population spreads on the neutral networks. With an appropriate redundant mapping with extensive neutral networks, it is conceivable that the population can be spread out over the entire genotype space. In this case, the population still maintains the ability to quickly search the space for a new optimal phenotype.

One of the anonymous reviewers suggested the interesting possibility to use the dynamics of the diversity of the population as an analytical tool to test for the existence of neutral networks. Because with real organisms the introduction of single mutations is technically hard the statistical methods described above cannot be applied. However, the diversity may be measured with standard techniques from population genetics. Thus, by looking at the dynamics of the population after a change has occurred one may be able to make inferences about the extent of the neutral networks the organisms evolve on.

6 Conclusion

The use of redundant mappings provides a number of benefits to the population. If the mapping between phenotype and fitness is unknown it pays to use an appropriate redundant mapping because it increases the accessibility between phenotypes. The smoother the landscape the easier it is to climb to the top of the landscape. For smooth unimodal landscapes one can simply use a hill climber to solve the problem. However, if the mapping from phenotype space to fitness is unknown then one gets a very rugged landscape in which it is very difficult to locate the optimum. With a redundant mapping which possesses highly intertwined networks the climb to the top may become possible due to the added connectivity between phenotypes.

Another benefit is that individuals are able to maintain higher fitness values in the presence of high mutation rates because a large fraction of all mutations are neutral. In addition, the population is quickly able to locate a new optimum should the environment change as redundancy in the genotype-phenotype mapping is a natural form of diversity maintenance. Convergence to a single genotype is avoided and the population is still able to perform a random search should this be necessary. Therefore, the use of redundant mappings generally increase the evolvability of the population employing such a mapping.

Acknowledgment

Part of this work was supported by the Santa Fe Institute, Santa Fe, NM. We thank Richard Watson, Brandeis University, Volen Center for Complex Systems, and Susan Ptak from Stanford University, Department of Biological Sciences, Stanford, for discussions and helpful comments.

Tracy Teal from UCLA Life Sciences, Los Angeles, also provided helpful comments on an earlier version of this paper. To evaluate the mappings in an evolutionary setting we have used GALOPPS [10].

References

[1] P. J. Angeline. Genetic programming and emergent intelligence. In K. E. Kinnear, Jr., editor, *Advances in Genetic Programming*, pages 75–97, Cambridge, Massachusetts, 1994. The MIT Press.

[2] W. Banzhaf. Genotype-phenotype-mapping and neutral variation – a case study in genetic programming. In Y. Davidor, H.-P. Schwefel, and R. Männer, editors, *Parallel Problem Solving from Nature – PPSN III. Proceedings of the International Conference on Evolutionary Computation*, pages 322–332, Berlin, 1994. Springer-Verlag.

[3] W. Banzhaf, P. Nordin, R. E. Keller, and F. D. Francone. *Genetic Programming - An Introduction: On The Automatic Evolution of Computer Programs and Its Applications.* Morgan Kaufmann Publishers, San Francisco, California, 1998.

[4] L. Barnett. Tangled webs: Evolutionary dynamics on fitness landscapes with neutrality. Master's thesis, MSc Dissertation. School of Cognitive Sciences, University of East Sussex, Brighton, August 1995.

[5] S. Bullock. Smooth operator? understanding and visualising mutation bias. In *Submission to the 6th European Conference on Artificial Life*, Prague, Republic, 2001.

[6] M. Ebner. On the search space of genetic programming and its relation to nature's search space. In *Proceedings of the 1999 Congress on Evolutionary Computation, Volume 2, Washington, D.C., July 6-9*, pages 1357–1361. IEEE Press, 1999.

[7] M. Ebner, P. Langguth, J. Albert, M. Shackleton, and R. Shipman. On neutral networks and evolvability. In *Proceedings of the 2001 Congress on Evolutionary Computation, COEX Center, Seoul, Korea.* IEEE Press, 2001.

[8] A. Gaspar and P. Collard. There is alife beyond convergence: Using a dual sharing to adapt in time dependent optimization. In *Proceedings of the 1999 Congress on Evolutionary Computation*, volume 3, pages 1867–1874, Mayflower Hotel, Washington D.C., USA, 6-9 July 1999. IEEE Press.

[9] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley Publishing Company, Reading, Massachusetts, 1989.

[10] E. D. Goodman. *An Introduction to GALOPPS. The "Genetic ALgorithm Optimized for Portability and Parallelism" System. Release 3.2. Technical Report # 96-07-01.* Michigan State University, July 1996.

[11] J. J. Grefenstette. Evolvability in dynamic fitness landscapes: A genetic algorithm approach. In *Proceedings of the 1999 Congress on Evolutionary Computation*, volume 3, pages 2031–2038, Mayflower Hotel, Washington D.C., USA, 6-9 July 1999. IEEE Press.

[12] F. Gruau. Genetic synthesis of modular neural networks. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms, University of Illinois at Urbana-Champaign, July 17-21, 1993*, pages 318–325, San Mateo, California, 1993. Morgan Kaufmann Publishers.

[13] F. Gruau. Genetic micro programming if neural networks. In K. E. Kinnear, Jr., editor, *Advances in Genetic Programming*, pages 495–518, Cambridge, Massachusetts, 1994. The MIT Press.

[14] W. E. Hart, T. E. Kammeyer, and R. K. Belew. The role of development in genetic algorithms. In D. Whitley and M. Vose, editors, *Foundations of Genetic Algorithms III*, pages 315–332, San Francisco, 1994. Morgan Kaufmann Publishers. Also available as Technical Report Number CS94-394, University of California, Computer Science and Engineering, San Diego, La Jolla.

[15] J. H. Holland. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence.* The MIT Press, Cambridge, Massachusetts, 1992.

[16] M. A. Huynen. Exploring phenotype space through neutral evolution. *Journal of Molecular Evolution*, 43:165–169, 1996.

[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] B. A. Julstrom. Redundant genetic encodings may not be harmful. In W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, editors, *GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference, Volume 1, July 13-17, 1999, Orlando, Florida*, page 791, San Francisco, California, 1999. Morgan Kaufmann Publishers.

[19] H. Kargupta. The genetic code and the genome representation. In *In Workshop "Gene Expression: The Missing Link in Evolutionary Computation" at the Genetic and Evolutionary Computation Conference (GECCO-2000), July 8, Las Vegas, Nevada*, 2000.

[20] S. A. Kauffman. *The Origins of Order. Self-Organization and Selection in Evolution.* Oxford University Press, Oxford, 1993.

[21] R. E. Keller and W. Banzhaf. The evolution of genetic code in genetic programming. In W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, editors, *GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference, Volume 1, July 13-17, 1999, Orlando, Florida*, pages 1077–1082, San Francisco, California, 1999. Morgan Kaufmann Publishers.

[22] M. Kimura. *Population Genetics, Molecular Evolution, and the Neutral Theory: Selected Papers.* The University of Chicago Press, Chicago, 1994.

[23] J. R. Koza. *Genetic Programming. On the Programming of Computers by Means of Natural Selection.* The MIT Press, Cambridge, Massachusetts, 1992.

[24] J. R. Koza. *Genetic Programming II. Automatic Discovery of Reusable Programs.* The MIT Press, Cambridge, Massachusetts, 1994.

[25] J. R. Koza, F. H. Bennett III, D. Andre, and M. A. Keane. *Genetic Programming III. Darwinian Invention and Problem Solving.* Morgan Kaufmann Publishers, 1999.

[26] M. Land and R. K. Belew. No perfect two-state cellular automata for density classification exists. *Physical Review Letters*, 74(25):5148–5150, June 1995.

[27] W. B. Langdon and R. Poli. Fitness causes bloat: Mutation. In W. Banzhaf, R. Poli, M. Schoenauer, and T. C. Fogarty, editors, *Genetic Programming: Proceedings of the First European Workshop, EuroGP'98, Paris, France, April 14-15*, pages 37–48, Berlin, 1998. Springer-Verlag.

[28] W. B. Langdon and R. Poli. Genetic programming bloat with dynamic fitness. In W. Banzhaf, R. Poli, M. Schoenauer, and T. C. Fogarty, editors, *Genetic Programming: Proceedings of the First European Workshop, EuroGP'98, Paris, France, April 14-15*, pages 97–112, Berlin, 1998. Springer-Verlag.

[29] J. R. Levenick. Swappers: Introns promote flexibility, diversity and invention. In W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, editors, *GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference, Volume 1, July 13-17, 1999, Orlando, Florida*, pages 361–368, San Francisco, California, 1999. Morgan Kaufmann Publishers.

[30] W. Liles and K. De Jong. The usefulness of tag bits in changing environments. In *Proceedings of the 1999 Congress on Evolutionary Computation*, volume 3, pages 2054–2059, Mayflower Hotel, Washington D.C., USA, 6-9 July 1999. IEEE Press.

[31] M. Mitchell. *An Introduction to Genetic Algorithms.* The MIT Press, Cambridge, Massachusetts, 1996.

[32] R. W. Morrison and K. A. De Jong. A test problem generator for non-stationary environments. In *Proceedings of the 1999 Congress on Evolutionary Computation*, volume 3, pages 2047–2053, Mayflower Hotel, Washington D.C., USA, 6-9 July 1999. IEEE Press.

[33] M. E. J. Newman and R. Engelhardt. Effects of selective neutrality on the evolution of molecular species. *Proc. R. Soc. Lond. B*, 265:1333–1338, 1998.

[34] P. Schuster. Extended molecular evolutionary biology: Artificial life bridging the gap between chemistry and biology. In C. G. Langton, editor, *Artificial Life: An Overview*, pages 39–60, Cambridge, Massachusetts, 1995. The MIT Press.

[35] M. Shackleton, R. Shipman, and M. Ebner. An investigation of redundant genotype-phenotype mappings and their role in evolutionary search. In *Proceedings of the 2000 Congress on Evolutionary Computation, San Diego Marriott Hotel, La Jolla, CA, Volume 1*, pages 493–500. IEEE Press, 2000.

[36] R. Shipman. Genetic redundancy: Desirable or problematic for evolutionary adaptation? In A. Dobnikar, N. C. Steele, D. W. Pearson, and R. F. Albrecht, editors, *4th International Conference on Artificial Neural Networks and Genetic Algorithms (ICANNGA '99)*, pages 337–344, New York, April 1999. Springer-Verlag.

[37] R. Shipman, M. Shackleton, M. Ebner, and R. Watson. Neutral search spaces for artificial evolution: A lesson from life. In M. A. Bedau, S. Rasmussen, J. S. McCaskill, and N. H. Packard, editors, *Artificial Life: Proceedings of the Seventh International Conference on Artificial Life.* MIT Press, August 2000.

[38] F. Siegmund. Einfluß der Kommunikationstopologie auf massiv parallele genetische Algorithmen. Master's thesis, Friedrich-Alexander-Universität Erlangen-Nürnberg, Aug 1992.

[39] M. Sipper. *Evolution of Parallel Cellular Machines: The Cellular Programming Approach.* Springer-Verlag, Berlin, 1997.

[40] P. W. H. Smith and K. Harries. Code growth, explicitly defined introns, and alternative selection schemes. *Evolutionary Computation*, 6(4):339–360, 1999.

[41] S. A. Stanhope and J. M. Daida. (1+1) genetic algorithm fitness dynamics in a changing environment. In *Proceedings of the 1999 Congress on Evolutionary Computation*, volume 3, pages 1851–1858, Mayflower Hotel, Washington D.C., USA, 6-9 July 1999. IEEE Press.

[42] K. Trojanowski and Z. Michalewicz. Searching for optima in non-stationary environments. In *Proceedings of the 1999 Congress on Evolutionary Computation*, volume 3, pages 1843–1850, Mayflower Hotel, Washington D.C., USA, 6-9 July 1999. IEEE Press.

[43] E. van Nimwegen, J. P. Crutchfield, and M. Huynen. Neutral evolution of mutational robustness. *Proc. Natl. Acad. Sci. USA*, 96:9716–9720, August 1999.

[44] G. P. Wagner and L. Altenberg. Complex adaptations and the evolution of evolvability. *Evolution*, 50(3):967–976, 1996.

[45] R. A. Watson. Analysis of recombinative algorithms on a non-separable building-block problem. In *Foundations of Genetic Algorithms 2000*, 2001.

[46] R. A. Watson, G. S. Hornby, and J. B. Pollack. Modeling building-block interdependency. In *Parallel Problem Solving from Nature - PPSN V*, pages 97–106, Berlin, 1998. Springer-Verlag.

[47] R. A. Watson and J. B. Pollack. Hierarchically-consistent test problems for genetic algorithms. In *Proceedings of the 1999 Congress on Evolutionary Computation*, pages 1406–1413, Mayflower Hotel, Washington D.C., USA, 6-9 July 1999. IEEE Press.

[48] R. A. Watson and J. B. Pollack. Hierarchically consistent test problems for genetic algorithms: Summary and additional results. In *GECCO-99 Late Breaking Papers*, pages 292–297, 1999.

[49] R. A. Watson and J. B. Pollack. Recombination without respect: Schema combination and disruption in genetic algorithm crossover. In Whitley et al., editor, *Proceedings of the 2000 Genetic and Evolutionary Computation Conference.* Morgan Kaufmann, 2000.

[50] K. Weicker and N. Weicker. On evolution strategy optimization in dynamic environments. In *Proceedings of the 1999 Congress on Evolutionary Computation*, volume 3, pages 2039–2046, Mayflower Hotel, Washington D.C., USA, 6-9 July 1999. IEEE Press.

[51] M. Wineberg and F. Oppacher. The benefits of computing with introns. In J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, editors, *Genetic Programming 1996, Proceedings of the First Annual Conference, July 28-31, 1996, Stanford University*, pages 410–415, Cambridge, Massachusetts, 1996. The MIT Press.

[52] S. Wolfram. Statistical mechanics of cellular automata. *Reviews of Modern Physics*, 55(3):601–644, July 1983.

[53] S. Wolfram. Cellular automata as models of complexity. *Nature*, 311(4):419–424, October 1984.

[54] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, April 1997.

[55] G. Woodcock and P. G. Higgs. Population evolution in a single peak fitness landscape. how high are the clouds? In F. Morán, A. Moreno, J. J. Merelo, and P. Chacón, editors, *Advances in Artificial Life. Proceedings of the Third European Conference on Artificial Life, Granada, Spain, June 4-6*, pages 148–157, Berlin, 1995. Springer-Verlag.

[56] T. Yu and J. Miller. Neutrality and the evolvability of boolean function landscape. In J. Miller, M. Tomassini, P. L. Lanzi, C. Ryan, A. G. B. Tettamanzi, and W. B. Langdon, editors, *Genetic Programming: Proceedings of the Fourth European Conference, EuroGP 2001, Lake Como, Italy, April 18-20*, pages 204–217, Berlin, 2001. Springer-Verlag.