Population Flow on Fitness Landscapes

Wim Hordijk
bkhordyk@cs.few.eur.nl

Supervisor:
Bernard Manderick

August 1994

Erasmus University Rotterdam
Faculty of Economics
Department of Computer Science

"We need a real theory relating the structure of rugged multipeaked fitness landscapes to the flow of a population upon those landscapes.

We do not yet have such a theory."

Stuart A. Kauffman

Contents

1 Introduction **1**
1.1 The goal of this thesis 2
1.2 The outline of the thesis 3
1.3 Acknowledgements 4

2 Fitness Landscapes **5**
2.1 The concept of fitness 5
2.1.1 Fitness in biology 5
2.1.2 Fitness in problem solving 6
2.1.3 The fitness function 7
2.2 Fitness landscapes 8
2.2.1 Bit strings and Hamming distance 8
2.2.2 The genotype space 8
2.2.3 The fitness landscape 9
2.2.4 The structure of a fitness landscape 9
2.3 The NK-model 10
2.3.1 The NK-model of epistatic interactions 10
2.3.2 Properties of the NK-model 12
2.4 Summary 13

3 Search Strategies and Performance **15**
3.1 Search strategies 15
3.1.1 Hillclimbing 15
3.1.2 Long jumps 18
3.1.3 Genetic Algorithm 18
3.1.4 Hybrid Genetic Algorithm 22
3.2 Performance measures 23
3.2.1 On-line performance 23
3.2.2 Off-line performance 24
3.2.3 Mean Hamming distance 24
3.3 Summary 25

4 The Structure of Fitness Landscapes **27**
4.1 The correlation structure . . . 27
4.1.1 Measuring correlation . . . 27
4.1.2 Time series analysis . . . 29
4.1.3 Handling other operators . . . 29
4.2 The Box-Jenkins approach . . . 30
4.3 The correlation structure of NK-Landscapes . . . 33
4.3.1 Results for point mutation . . . 34
4.3.2 Results for crossover . . . 40
4.3.3 Results for long jumps . . . 45
4.4 Conclusions . . . 46

5 Population Flow **47**
5.1 Experimental setup . . . 47
5.2 Evaluation by maximum fitness . . . 48
5.2.1 Smooth landscapes: K=0 . . . 48
5.2.2 Rugged landscapes: K=2, 5 . . . 50
5.2.3 Very rugged landscapes: K=25, 50 . . . 51
5.2.4 Completely random landscapes: K=99 . . . 51
5.3 Evaluation by on-line performance . . . 53
5.4 Evaluation by off-line performance . . . 55
5.5 Evaluation by mean Hamming distance . . . 55
5.6 Conclusions . . . 58
5.6.1 General conclusions . . . 58
5.6.2 Time scales in adaptation . . . 59
5.6.3 Some implications . . . 60
5.6.4 Summary . . . 62

6 The Usefulness of Recombination **63**
6.1 Crossover disruption . . . 63
6.1.1 Experimental setup . . . 64
6.1.2 Results . . . 64
6.2 Recombination and the location of optima . . . 68
6.2.1 Experimental setup . . . 68
6.2.2 Results . . . 69
6.3 Conclusions . . . 73

7 Conclusions and Further Research **75**
7.1 The structure of fitness landscapes . . . 75
7.2 Time scales in adaptation . . . 76
7.3 The usefulness of recombination . . . 77
7.4 Directions for further research . . . 77

A A Two-Sample Test for Means **79**

B The Height of Peaks in a Landscape **80**

C Used Software **83**

Chapter 1

Introduction

The last two or three decades, there has been an increasing interest in using an evolutionary approach to problem solving (see for example [BHS91, Hol92]). At the same time, biologists are beginning to consider evolution more and more as a combinatorial optimization problem, that is, a problem with a large but finite number of solutions. But although much progress is made with these new developments, evolution is still not fully understood.

The first real theory of evolution was put forward in 1859 by Darwin. His theory is based on variations between the members of a population, and the "*preservation of favourable variations and the rejection of injurious variations*", which he called *Natural Selection* [Dar59]. During the following decades, the causes of these variations, unknown to Darwin himself, were gradually laid bare. Every organism contains genetic material (the genotype) that determines the appearance of this organism (the phenotype). During reproduction, this genetic material is passed on to the offspring, but different genetic operators alter the genetic material of the offspring, causing it to be different from that of its parent(s). These genetic operators include crossover (exchanging parts of the genetic material of two parents) and mutation (small changes in the genetic material, for example caused by "copying errors").

The genetic material turned out to be stated in a "universal" genetic code, which was cracked a century after Darwin came up with his theory. So, an organism (the phenotype) can be represented by a genotype by means of this code. In fact, this is exactly what is done in evolutionary biology: the evolution of a population of organisms is considered as the evolution of a population of genotypes in a large, but finite space of all possible genotypes.

In the evolutionary approach to problem solving, Nature is imitated. Given a certain problem, a coding is used to represent the possible solutions for this problem in the form of genotypes. Starting with one or more randomly chosen genotypes, new generations of this population of genotypes are created by repeatedly applying one or more genetic operators (for example crossover and mutation) to these genotypes, in the hope that new genotypes are formed that represent increasingly better solutions for the given problem. Usually, some form of selection, which tends to keep only the better solutions (or, in fact, genotypes) in the population, is also applied. This way an optimal solution is searched for by an imitation of natural evolution, also in a large, but finite genotype space.

Both in Nature and problem solving, an individual (be it an organism or a solution for a problem) can be assigned a fitness. For now, consider this fitness as a measure of success (for example in survival or in solving a problem). So, to every possible genotype belongs a certain fitness. The distribution of these fitness values over the space of possible genotypes constitutes a fitness landscape. Imagine this as a landscape with hills and valleys, hilltops denoting high fitness, valleys denoting low fitness. (The concepts of fitness and fitness landscapes is explained in more detail in the next chapter.)

The evolution of a population of individuals (whether organisms or solutions) can now be visualized as a population of genotypes adapting on a fitness landscape, in search for the highest peaks. Knowing the structure of this underlying fitness landscape can help a lot in understanding, interpreting, and perhaps predicting the evolution of such a population.

1.1 The goal of this thesis

Until now, little is known about how populations evolve, or adapt, on different kinds of fitness landscapes. The main goal of this thesis is therefore to gain more insight into the *population flow on fitness landscapes*, which hopefully contributes to a theory relating the structure of a fitness landscape to the flow of a population on it. Such a theory can help both in biology, for a better understanding of the principles of evolution, and in problem solving, for finding better ways to solve a difficult problem. Kauffman made a start in this direction, and a large part of this thesis depends on his work (see [Kau93]).

To reach the stated goal, the (global) structure of a fitness landscape has to be known first. Different procedures have been used to measure this structure ([Wei90, MWS91, Lip91]). A more complete statistical procedure, based on the one introduced in [Wei90], to determine and express this global structure is proposed (and applied) here.

Next, different search strategies (some of which are biologically inspired) are applied to different fitness landscapes, to gain some insight into population flow in general. The strategies are compared to each other by a couple of performance measures. Besides, the validity of the next statement made by Kauffman is assessed. He identifies *three natural time scales* in adaptation on rugged fitness landscapes:

1. Initially, fitter individuals (individuals having a higher fitness) are found faster by long jumps (jumps in the landscape over a long distance) than by local search. However, the waiting time to find such fitter individuals doubles each time one is found.

2. Therefore, in the midterm, adaptation finds nearby fitter individuals faster than distant fitter individuals and hence climbs a local hill in the landscape. But the rate of finding fitter nearby individuals first dwindles and then stops as a local optimum ("the top of a hill") is reached.

3. On the longer time scale, the process, before it can proceed, must await a successful long jump to a better hillside some distance away.

He states that "*this outline frames in the behavior of an adapting population when the rate of finding fitter individuals is low compared with fitness differentials*".

Finally, the usefulness of recombination (exchanging parts of two genotypes to form a new one) is examined more thoroughly, and the validity of a second statement made by Kauffman is assessed. He states that "*recombination is useless on uncorrelated landscapes but useful under two conditions: (1) when the high peaks are near one another and hence carry mutual information about their joint locations in the fitness landscape and (2) when parts of the evolving individuals are quasi-independent of one another and hence can be interchanged with modest chances that the recombined individual has the advantage of both parents*".

1.2 The outline of the thesis

The next two chapters introduce the basic concepts on which the rest of the thesis is based. First, Chapter 2 gives an introduction to the concepts of fitness and fitness landscapes, and the biological background from which all this is derived. Furthermore, it introduces the NK-model, which is a model for general fitness landscapes. This model is used throughout this thesis. Chapter 3 then introduces the different search strategies that are applied to landscapes generated by the NK-model. It also introduces some performance measures by which the strategies are compared.

In Chapter 4, a statistical procedure is proposed to determine and express the global structure of fitness landscapes. The results of applying this procedure to landscapes generated by the NK-model are also presented in this chapter. Next, Chapter 5 presents the results of applying the search strategies, introduced in Chapter 3, to fitness landscapes generated by the NK-model. The results are evaluated by the performance measures also introduced in Chapter 3. Besides, the validity of Kauffman's statement about the three time scales in adaptation, mentioned in Section 1.1, is assessed. The focus of Chapter 6 is on the usefulness of recombination. This reproduction strategy is examined more thoroughly, and Kauffman's statement about the usefulness of recombination (see section 1.1) is put to the test.

Finally,Chapter 7 summarizes the major conclusions reached in the previous chapters, followed by some directions for further research.

1.3 Acknowledgements

First of all I want to thank my supervisor Bernard Manderick. Without his invaluable comments, ideas and criticisms, this thesis would have got stuck somewhere halfway up the hillside. He made me see how important it is to always be careful in interpreting results, and how useful the vision of an "experienced eye" is.

Furthermore I want to say thanks to Rémon Sinnema for letting me use his software, for the fruitful discussions, and for sharing many thoughts, ideas and beers. Climbing the hill together is much more fun than doing it all alone!

Also thanks to *hathi* for going down only once while being pushed to the limit for more than two months.

Last, but not least, I especially want to thank my parents for smoothing, in many ways, the landscape I had to walk on for the past years.

Chapter 2

Fitness Landscapes

The notion of fitness and fitness landscapes comes from biology, where it is used as a framework for thinking about evolution. This approach has proved to be very useful. In the evolutionary approach to problem solving, this paradigm has also been adopted. It has become a central theme in the evolutionary sciences.

This chapter first introduces the concepts of fitness and fitness landscapes, and the biological background on which these concepts are based. Next, the NK-model is introduced. This model generates different kinds of fitness landscapes, and it is used throughout this thesis.

2.1 The concept of fitness

2.1.1 Fitness in biology

The term *fitness* originally stems from biology: "*In essence, the fitness of an individual depends on the likelihood that one individual, relative to other individuals in the population, will contribute its genetic information to the next generation. Fitness, then, includes the relative ability of an organism to survive, to mate successfully, and to reproduce, resulting in a new organism.*" [WH88]. This use of the term fitness is a direct consequence of the theory of Natural Selection, which states that better adapted individuals will on average leave more offspring than less adapted individuals.

This definition implies that the fitness of an individual can only be determined afterwards. But the main point is, that the fitness of an individual somehow denotes its chances of leaving offspring (pass on its genetic information), and that it is a measure of "how good" the individual is, relative to the other individuals in the population.

In biology, a distinction is made between the genotype and the phenotype of an organism. The *genotype* of an organism is the genetic make-up of this organism (the genetic information that is stored in the form of DNA in the chromosomes of every living cell). The *phenotype* is the appearance of the organism — the expression of the genotype. This means, the organism as it appears and interacts in the environment it finds itself in.

The genetic information is stated in a universal[1] genetic code that is represented by the four letters `A`, `T`, `C` and `G`. So, a phenotype can be represented by a genotype, which is an alternating sequence of four letters, something like `ATCCGTCGAA`. The exact sequence of these four letters determines the phenotypical expression[2].

In the process of reproduction, the genetic information that is passed on to the offspring is altered in some ways. Sometimes a "copying error" occurs, for example when a `C` is by mistake copied to a `T`. This is called *mutation*. When the reproduction is sexual, i.e. two parents are involved, the genetic information of the two parents is mingled, and a new genotype that is different from that of both parents is created. This is called *crossover*. It is by means of these variations that evolution is possible. Variations that are useful to an organism will, on average, be preserved, and variations that are harmful to an organism will, again on average, be rejected by the process of Natural Selection.

So, the fitness of an organism is assigned directly to the phenotype (according to how well it is adapted to survive and reproduce), but the evolution itself takes place at the level of the genotype. Genotypes that code for successful (well adapted) organisms will have a higher chance of being passed on than genotypes that code for unsuccessful (not well adapted) organisms. So, the genotypes are assigned a fitness indirectly.

2.1.2 Fitness in problem solving

In the evolutionary approach to problem solving, the distinction into genotype and phenotype is copied. Here, the phenotype is a solution for the given problem. This can be an integer, a graph, a permutation, or whatever. The genotype, then, is a *coding* for such a solution, just as the DNA of an organism is the coding for the appearance of this organism.

Take for example the problem of maximizing the function $f(x) = x^2$ over the integers in the range $[0, 31]$. The integers from 0 to 31 can be coded by their binary representation. Strings of length 5 will then be needed. The string `00000` codes for the integer 0, the string `00001` codes for the integer 1, and so on until `11111`, which codes for the integer 31. So, in this example a genotype is a string of length 5 consisting of `0`'s and `1`'s, and a phenotype is an integer which is a possible solution to the given problem.

Now every solution, or phenotype, can be assigned a fitness. In this case, the fitness is a measure of "how good" the phenotype is for the given problem, relative to other

[1]Universal means that the genetic information of every living creature on earth is stated in this code.

[2]This is a rather simplified view that can not be hold in real life. The environment also plays a major role in the development of an organism, but the simplified view is used in modelling evolution.

phenotypes. In the example above, the fitness of a phenotype is just its square. So, 0 has a fitness of $0^2 = 0$, 3 has a fitness of $3^2 = 9$, etc. It is easy to see that the phenotype 31 has the highest fitness of all possible phenotypes and thus is the optimum. Here too, the genotypes are assigned a fitness indirectly through their corresponding phenotypes. So, the genotype `00011`, which codes for the phenotype 3, has a fitness of 9.

Just as in Nature, different genetic operators can now be applied to the genotypes, making a form of evolution possible. The fitness of the corresponding phenotypes is used to simulate Natural Selection: high fitness means a high chance of being chosen to contribute the genetic material to the offspring, low fitness means a low chance of being chosen[3].

In the example above, it is rather straightforward what the phenotypes and genotypes are, and what their fitness is. But in general, this will not be the case. Most (real-world) problems will be more complex than the one above, and will not be solvable in an analytical way. Furthermore, a solution for a problem can be a graph, or a permutation, or an even more complex data structure, instead of just an integer. In this case, it will not immediately be obvious what the fitness of a solution is. To handle this problem of assigning a fitness to a solution, a fitness function is used.

2.1.3 The fitness function

A *fitness function* is a mathematical description of a certain problem. It is used to evaluate different solutions for this problem, just as Natural Selection "evaluates" organisms in Nature. The better a solution is for the given problem, relative to other solutions, the higher its fitness will be.

A fitness function takes as its input the coding (the genotype) of a possible solution, translates this genotype to the corresponding solution (the phenotype), applies this solution to the given problem and returns a fitness value according to "how good" the solution is for this problem. In the example above, the fitness function takes as input a string of length 5 consisting of `0`'s and `1`'s, considers this as the binary representation of an integer, and returns the square of this integer.

The difference between Nature and the evolutionary approach to problem solving is, that in Nature the fitness function is implicit (fitness is assigned by means of the selection process), while in problem solving the fitness function is explicit (to make the simulation of a selection process possible).

Now that the concepts of fitness, genotypes and phenotypes, and fitness functions are known, the concept of a fitness landscape can be explained.

[3] Genotypes have to be picked out and reproduced by some external force, usually a computer program, because they cannot do this themselves, of course.

2.2 Fitness landscapes

2.2.1 Bit strings and Hamming distance

To explain fitness landscapes, a notion of a *distance* between genotypes is needed. Genotypes are codings, and different codings can induce different distance measures. Also, often more than one distance measure, or metric, can be defined for one and the same coding.

Usually, a coding in the form of *bit strings* is used. Bit strings are strings consisting only of `0`'s and `1`'s, like `0110100111`. Bit strings have some advantages over other codings. First of all, genetic operators like crossover and mutation are easy to apply in such a way that the results are bit strings again (which, of course, is necessary). Second, bit strings can be implemented very easily in computer programs. Third, a very natural and widely used metric is defined for bit strings: the Hamming distance.

The *Hamming distance* between two bit strings is defined as the number of corresponding positions in these bit strings where the bits have a different value. So, the distance between `010` and `100` is two, because the first and second positions have different values. A *normalized* Hamming distance can be defined by dividing the Hamming distance between two bit strings by the length of these bit strings. This way, the distance measure is independent of the length of the bit strings. A normalized Hamming distance of 0.5, for example, means that half the bits of two bits string have a different value.

Throughout this thesis, bit strings are used as a coding, and the Hamming distance is used as metric.

2.2.2 The genotype space

If the possible solutions for a given problem are encoded by some form of genotype, then the problem space (the abstract space of all possible solutions) can also be represented by a genotype space. A *genotype space* is the (mostly high dimensional) space in which each point represents one genotype and is next to all other points that have a distance of one from this point (according to some appropriate metric). All the points at distance one are called the *neighbors* of this first point, and together they form a *neighborhood*. (Note that this genotype space is a *discrete* space.)

The next example will make things more clear. Consider as genotypes bit strings of length 3. The total number of bit strings of this length is $2^3 = 8$. With the Hamming distance (see section 2.2.1) as metric, every bit string of length three has exactly three neighbors, namely those bit strings that differ in one of the three bits. The corresponding genotype space is shown in Figure 2.1 (ignore the figures between parentheses for now). Every point in the space represents one genotype and has exactly three neighbors, each of which differs in the value of one bit.

2.2.3 The fitness landscape

Now every genotype will have a certain fitness, which is determined by some fitness function (see Section 2.1.3). The *fitness landscape* is then constructed by assigning the fitness values of the genotypes to the corresponding points in the genotype space. This can be envisioned as giving each point in the genotype space a "height" according to its fitness. This way, a more or less "mountainous" landscape is formed, where the highest peaks designate the best solutions. A local optimum, or peak, in such a landscape is defined as a point that has a higher fitness than all its neighbors. Note (again) that this landscape is discrete.

In the genotype space of Figure 2.1, each point has been assigned a value from 1 to 8 at random, which denotes its fitness (shown in parentheses), thus making it a fitness landscape. It can be seen that every point, except `100` and `001`, has at least one neighbor with a higher fitness. For the two exceptions, designated by a dashed circle, all neighbors have a lower fitness, and thus these two points are local optima in this landscape.

So, this particular fitness landscape contains two peaks. Note that when a different metric is chosen, the landscape can change too, because other points are defined as being neighbors. Hence, a point that is a local optimum in one landscape is not necessarily a local optimum in another landscape, because it can have other neighbors.

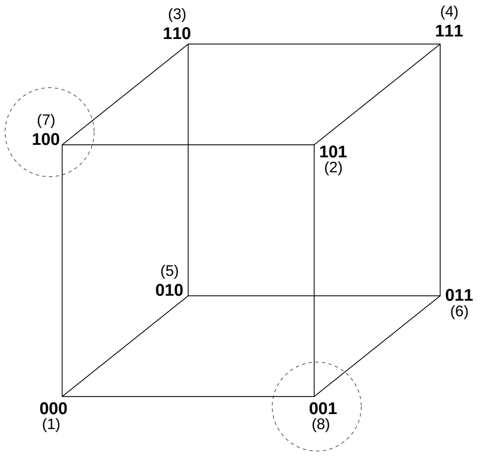

Figure 2.1: The fitness landscape for bit strings of length 3. Every point on the cube represents a genotype, and is connected to its three neighbors. Each point has been assigned a fitness at random, ranging from 1 (low) to 8 (high). These fitness values are shown in parentheses. The two local optima are designated by a dashed circle.

2.2.4 The structure of a fitness landscape

Summarizing, a fitness landscape is defined by three things:

1. A coding for the possible solutions for a problem (the genotypes)

2. A metric that defines which genotypes are neighbors

3. A fitness function that defines the fitness of the genotypes

The first two items define the genotype space. Adding the third item gives the fitness landscape. If one of these three items changes, the landscape will change as well. So, in general, there is not a *unique* fitness landscape for a given problem, and the structure of the landscape depends on the above three items.

The structure of a landscape incorporates many things, like the dimensionality (the number of neighbors each point in the genotype space has), the number of peaks, the "steepness" of the hillsides, the relative height of the peaks, etc.

A landscape where the average difference in fitness between neighboring points is relatively small, is called *smooth*. On such a landscape it will be easy to find good optima: local information about the landscape can be used effectively to direct the search. A landscape with a relatively large average fitness difference between neighbors, is called *rugged*. On such a landscape it will be difficult to find good optima: local information becomes less valuable.

So, the (global) structure of a landscape can range from very smooth to very rugged. One way to mathematically express this global structure of a landscape is by its *correlation structure*. What this means and how this is measured is explained in Chapter 4.

2.3 The NK-model

The structure of a fitness landscape depends on the underlying problem. But a theory about population flow should be independent of that. It would therefore be convenient to have a problem-independent fitness landscape. Kauffman introduced a model to generate such landscapes: the *NK-model* [Kau89]. The landscapes that result from this model (hereafter called *NK-landscapes*) can be tuned from smooth to rugged. The NK-model turned out to be a good model for a wide range of problems. Therefore it is used throughout this thesis for modelling general fitness landscapes.

2.3.1 The NK-model of epistatic interactions

The NK-model, of course, incorporates the three items that define a fitness landscape. As genotypes, bit strings of length N are used. As metric, the Hamming distance is taken (see Section 2.2.1). The fitness function is more complicated, and will be explained next.

Suppose every bit b_i ($i = 1, \ldots, N$) in the bit string $\underline{b}$ is assigned a fitness of its own. The fitness of a bit, however, does not only depend on the value (0 or 1) of this specific bit, but also on the value of K other bits in the same bit string ($0 \leq K \leq N-1$). These dependencies are called *epistatic interactions*. Thus the two main parameters of the NK-model are the number of bits, N, and the number of other bits K which epistatically influence the fitness contribution of each bit.

So, the fitness contribution of one bit depends on the value of $K + 1$ bits (itself and K others), giving rise to a total of 2^{K+1} possibilities. Since, in general, it is not known what the effects of these epistatic interactions are, they are *modelled* by assigning to each of the 2^{K+1} possibilities at random a fitness value drawn from the Uniform distribution between 0.0 and 1.0. Therefore, the fitness contribution w_i of bit b_i is specified by a list of random decimals between 0.0 and 1.0, with 2^{K+1} entries. This procedure is repeated for every bit b_i, $i = 1, \ldots, N$ in the bit string $\underline{b}$.

Having assigned the fitness contributions for every bit in the string, the fitness of the entire bit string, or genotype, is now defined as the average of the contributions of all the bits:

$$W = \frac{1}{N} \sum_{i=1}^{N} w_i$$

Table 2.1 gives an example (taken from [Kau93]) with N=3 and K=2. In this example, each bit depends on all other bits in the bit string. The fitness contributions in the fourth, fifth and sixth column are drawn at random. The total fitness of the genotype is calculated as the average of the fitness contribution of all bits in the string, and is given in the last column. The corresponding fitness landscape is shown in Figure 2.2.

value of bit			fitness contribution			total fitness
1	2	3	w_1	w_2	w_3	$W = \frac{1}{N}\sum_{i=1}^{N} w_i$
0	0	0	0.6	0.3	0.5	0.47
0	0	1	0.1	0.5	0.9	0.50
0	1	0	0.4	0.8	0.1	0.43
0	1	1	0.3	0.5	0.8	0.53
1	0	0	0.9	0.9	0.7	0.83
1	0	1	0.7	0.2	0.3	0.40
1	1	0	0.6	0.7	0.6	0.63
1	1	1	0.7	0.9	0.5	0.70

Table 2.1: Assignment of fitness values to each of the three bits with random values for each of the $2^{K+1} = 8$ possible situations. The total fitness of each genotype is the average of the three fitness contributions. Example taken from [Kau93].

One further aspect of the NK-model characterizes how the K epistatic interactions for each bit are chosen. Generally, this is done in one of two ways.

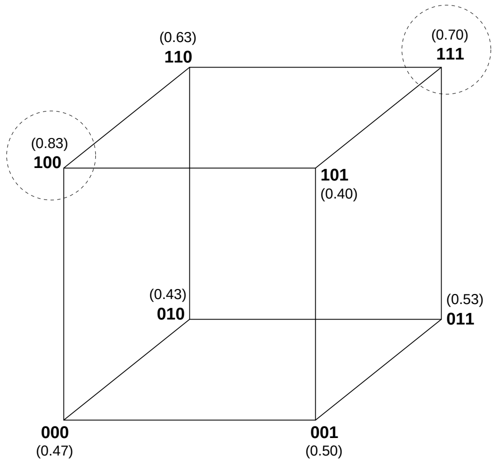

Figure 2.2: The fitness landscape corresponding to the example in Table 2.1. The fitness values are shown between parentheses. There are two local optima, which are designated by a dashed circle.

The first way is by choosing them at random from among the other $N-1$ bits. This is called *random interactions*. It is important to note that no reciprocity in epistatic influence is assumed. This means that if the fitness of bit b_i depends on bit b_j, it is not necessary that the reverse also holds. So, the epistatic interactions for a bit are determined independent of the other bits.

The second way is by choosing the K neighboring bits as epistatic interactions. The $K/2$ bits on each side of a bit will influence the fitness of this bit. This is called *nearest neighbor interactions*. To make this possible, *periodic boundary conditions* are taken into account. This means that the bit string is considered as being circular, so the first and the last bit are each others neighbors. Note that for K=0 and $K = N-1$, there is no difference between the two sorts of interactions. In the first case, the fitness of each bit depends only on its own value, and in the second case, the fitness of each bit depends on the value of all the bits in the string.

2.3.2 Properties of the NK-model

The main property of the NK-model, the property for which the model was formulated in the first place, is that the corresponding landscape can be tuned from smooth to rugged by changing the parameter K, relative to N. In the case of K=0, there are no epistatic interactions, and for each bit, by chance, either the value `0` or the value `1` makes the higher fitness contribution. Therefore, there is one genotype having the fitter value at each position which is the global optimum. Furthermore, every other genotype can be sequentially changed to the global optimum by successive flipping of each bit which has the less favored value to the more favored value. The landscape for K=0 is very smooth: neighboring genotypes do not differ much in their fitness values, and there is one (global) peak.

Increasing K introduces *conflicting constraints* among the different bits, and causes the landscape to become more rugged, because the complexity of the model increases. The case of $K = N - 1$ corresponds to a fully random landscape. Changing the value of only one bit causes a change in the fitness of all bits, because the fitness of each bit depends also on all other bits. Each bit now has a different (random) fitness, and therefore the fitness of the entire string changes to a completely random value. So, neighboring genotypes have very different fitnesses, and the landscape will be extremely rugged.

Kauffman has investigated the properties of the NK-model extensively. A summary of the most important conclusions is as follows [Kau93]:

- Almost all features of the fitness landscape depend entirely on N and K, making the NK-model a very simple but effective tool for investigation. Also, according to Kauffman, the features of the landscape do not depend on the type of interactions (random or nearest neighbor), nor on the type of distribution that is used to assign the random fitness contributions to each bit.

- When K is proportional to N, a *complexity catastrophe* sets in as N increases: attainable optima become ever more "mediocre", or typical of the entire landscape. When K remains small as N increases, this complexity catastrophe does not set in; hence low epistatic interactions are a sufficient "construction requirement" in complex systems in order to adapt on "good" fitness landscapes with high accessible optima.

- In an adaptive search, the time to find a fitter individual doubles every time such an individual is found.

- When a constant mutation rate (the rate at which bits "spontaneously" change their value) is assumed, an *error catastrophe* sets in as N increases: the ability of selection to hold an adapting population at a local optimum ultimately fails, and the population "melts" and flows down the hillside to drift neutrally through wide regions of the landscape.

2.4 Summary

The evolution of a population of individuals, whether real organisms or solutions for some optimization problem, can be modelled by an adapting population of genotypes on a fitness landscape. The genotype of an individual is its genetic coding, which determines the phenotype, the actual appearance of the individual. A fitness landscape, then, is the space of all possible genotypes with some neighborhood relation, where every genotype is assigned a fitness by means of a fitness function. The fitness of an individual denotes its relative success in leaving offspring, that is, relative to other individuals. A fitness function can be implicit (the fitness is determined by a selection process) or explicit (the fitness is used to simulate selection).

The NK-model is a useful model to generate fitness landscapes of which the global structure can be tuned from smooth (small differences in fitness between neighboring genotypes) to very rugged (large differences in fitness between neighboring genotypes) by changing the parameter K (the richness of epistatic interactions) relative to N (the length of the genotypes). Since Kauffman already showed what happens when N varies, given a (relative) value of K, and since the main interest in this thesis is what happens with populations on landscapes that differ in ruggedness, the value of N is fixed in all experiments, and K is varied relative to N.

Having introduced the concept of fitness landscapes, and a model to generates such landscapes, there still is no evolution, or population flow. To let this happen, some kind of search process has to take place on these landscapes. The next chapter introduces some search strategies that perform such processes on fitness landscapes.

Chapter 3

Search Strategies and Performance

On the one hand, evolutionary search strategies are used more and more to solve complex problems. On the other hand, evolution is considered more and more as a search process in a large, but finite space of possible solutions. This chapter introduces some search strategies that are applied to different kinds of fitness landscapes (see Chapter 2). The strategies all perform an adaptive search on these landscapes. Comparing the performances of the different strategies can give insight into what types of strategies work well on what types of landscapes, but also into the principles of evolution itself. Some performance measures are introduced as well, by which the strategies are evaluated.

3.1 Search strategies

The search strategies that are applied to different fitness landscapes are various implementations of the following four search methods:

- Hillclimbing
- Long jumps
- Genetic Algorithm
- Hybrid Genetic Algorithm

In this section, these four methods are introduced one by one. Their weaknesses and strengths are discussed, and the exact implementations that are used are given as well. With all strategies, it is assumed that the points (genotypes) in the landscape to which these strategies are applied, are bit strings.

3.1.1 Hillclimbing

Hillclimbing is a general, local search strategy that can be applied to a multitude of problems. The idea is to start at a randomly chosen point in the landscape, and walk via fitter neighbors to a nearby hilltop. If this procedure is repeated a couple of times, it is called *iterated hillclimbing.*

Basically, there are three forms of hillclimbing (steepest ascent, next ascent, and random ascent), which are all variants of the following general algorithm:

Hillclimbing

1. Choose a bit string at random. Call this string *current-hilltop*.
2. Choose a fitter neighboring string of *current-hilltop* by some criterion.
3. If a fitter neighbor could be found, then set *current-hilltop* to this new string, and return to step 2 with this new *current-hilltop*.
4. If no fitter neighbor could be found, then return the fitness of the *current-hilltop*.

With iterated hillclimbing, this procedure is restarted every time a local optimum has been found (that is, not fitter neighbor could be found), until a preset number of function evaluations has been reached. The local optima that are found during the search are saved, and in the end the best optimum found is returned.

The three forms of hillclimbing differ in the criterion by which a fitter neighboring string is chosen. These criteria are as follows [FM93]:

Steepest ascent hillclimbing: Systematically flip all bits in the string, recording the fitnesses of the resulting strings. Choose the string that gives the highest increase in fitness.

Next ascent hillclimbing: Flip single bits from left to right, until a neighbor is found that gives an increase in fitness. Choose this string as fitter neighbor. At the following step, however, continue flipping bits after the point at which the last fitness increase was found.

Random ascent hillclimbing: Flip bits at random, until a neighbor is found that gives an increase in fitness. Choose this string as fitter neighbor.

Note that the first two algorithms can be performed in an iterated way, because the bits are flipped systematically, so it is known when a local optimum has been reached. Random ascent hillclimbing, however, just keeps flipping bits at random, so it is never known whether a local optimum has been reached yet.

Hillclimbing is a very general search strategy that is often used as a "benchmark" for other search strategies. A search strategy should at least perform as well as hillclimbing. But in comparing other search strategies with hillclimbing, "*it matters* which *type of hillclimbing algorithm is used*" [FM93]. Therefore, two different hillclimbing variants will be used here.

The first hillclimbing variant is based on random ascent hillclimbing. Random ascent hillclimbing appears to be a very strong algorithm for some specially designed fitness landscapes, "*but [it] will have trouble with any function with local optima*" [FM93]. Therefore, an "extended version" is implemented here: *Random ascent hillclimbing with memory.* The algorithm "remembers" which bits it already has tried, and so it will know when it is at a local optimum. This way, the algorithm can be used in an iterative way too. The bits that are flipped are chosen at random, but without repetition (of course, every time a fitter neighbor has been found, all the bits can be chosen again). The implementation of this hillclimbing variant is as follows:

Random ascent hillclimbing with memory (RAHCM)

1. Set *best-evaluated* to 0.
2. Choose a bit string at random. Call this string *current-hilltop*. If the fitness of *current-hilltop* is higher than *best-evaluated*, then set *best-evaluated* to this fitness.
3. Choose a bit from *current-hilltop* at random, without repetition, and flip it. If this leads to an increase in fitness, then set *current-hilltop* to the resulting string, otherwise, repeat step 3. If the fitness of the new *current-hilltop* is higher than *best-evaluated*, then set *best-evaluated* to this fitness. Go to step 3 with the new *current-hilltop*, and forget all the bits that have been flipped so far.
4. If all bits of the *current-hilltop* have already been flipped once and no increase in fitness was found, then go to step 2.
5. When a set number of function evaluations has been performed, return *best-evaluated*.

The second hillclimbing variant combines elements of both steepest ascent and random ascent hillclimbing. Just as in steepest ascent hillclimbing, the fitness of all neighbors are calculated and stored. But where in random ascent a *bit* is chosen at random, in this variant a *fitter neighbor* is chosen at random. This is repeated until no fitter neighbors exist, and thus a local optimum has been reached. This variant will be called *Random neighbor ascent hillclimbing*, to emphasize that a fitter neighbor is chosen at random instead of just a bit. The implementation of this hillclimbing variant is as follows:

Random neighbor ascent hillclimbing (RNAHC)

1. Set *best-evaluated* to 0.
2. Choose a bit string at random. Call this string *current-hilltop*. If the fitness of *current-hilltop* is higher than *best-evaluated*, then set *best-evaluated* to this fitness.
3. Systematically flip each bit in the string from left to right, recording the strings that lead to an increase in fitness.

4. If there are strings that lead to an increase in fitness, then choose one of them at random and set *current-hilltop* to this string, otherwise go to step 2. If the fitness of the new *current-hilltop* is higher than *best-evaluated*, then set *best-evaluated* to this fitness. Go to step 3 with the new *current-hilltop*.

5. When a set number of function evaluations has been performed, return *best-evaluated*.

So, this algorithm is also used in an iterative way. This algorithm was used by Kauffman for examining the properties of NK-landscapes (see Chapter 2). It is assumed that he based his statement about the three time scales in adaptation (see Section 1.1) on the results obtained with this hillclimbing variant.

3.1.2 Long jumps

With long jumps, not just one bit is flipped, but a lot of bits are flipped at one step. This means that an individual jumps a long distance (in terms of Hamming distance) across the fitness landscape. Long jumps are implemented as follows:

Long jumps

1. Initialize a population of bit strings at random.

2. For each bit string in the population, make a long jump by systematically flipping each bit in the string with probability 0.5. If the resulting string has a higher fitness, then replace the old string with the new string, otherwise, keep the old string in the population.

3. Repeat step 2 for a set number of function evaluations.

Since every bit in a string is flipped with probability 0.5, it comes effectively down to just trying random strings to see if they are better. So, there is no direction in the search whatsoever. The only restriction is that only strings that are better than the previous one are allowed to enter the population.

Note that the algorithm can not be used in an iterative way, because it is never known when a local optimum has been reached (the immediate neighbors of a bit string are not evaluated). Therefore, the algorithm uses a population of searchers, but never starts anew. A population size of 50 is taken for all experiments.

3.1.3 Genetic Algorithm

A Genetic Algorithm (GA, see [Hol92, Gol89]) simulates natural evolution by repeatedly applying three operators to a population of genotypes: *selection*, *crossover* and *mutation*. The operators can be implemented in various ways, but here only the variants that are used are explained.

First, an initial population of genotypes (in the from of bit strings) is created at random. Each genotype in the population is assigned a fitness which is determined by some fitness function (see section 2.1.3). Next, new generations are created by repeatedly applying the three operators.

Selection
A new population is created by selecting at random genotypes from the old population, where the relative fitness of each genotype (relative to the fitness of the other genotypes in the population) determines its probability of being selected. So, genotypes with a high relative fitness have a higher probability of being selected than genotypes with a low relative fitness. On average, some (relatively fit) genotypes will be selected more than once, while some other (relatively unfit-fit) genotypes will not be selected at all. This selecting of genotypes is repeated until the new population is as large as the old one.

The selection mechanism that is used in the experiments, is called *Deterministic tournament selection*. This mechanism is implemented as follows:

Deterministic tournament selection

1. Choose s genotypes at random from the old population without repetition. s is the tournament size.

2. Take the fittest genotype of the s selected ones and place it in the new population.

3. Repeat steps 1 and 2 until the new population is as large as the old one.

This selection mechanism can be seen as random individuals in the population playing a tournament, and the most fit individual in this tournament wins, and is allowed to contribute its genetic information to the next generation. The tournament size s can be used to vary the selection pressure.

Crossover
Start by taking the first pair of genotypes from the new population as parents. With a certain chance p_c (called the *crossover rate*) exchange some parts of the genetic information of these two parents, thus creating two children. These children replace their parents in the population. Repeat this procedure for every next pair in the population.

The crossover rate p_c is a number between 0 and 1, determining the probability that this exchange of genetic information actually happens for a pair of parents. In practice, a rate somewhere between 0.6 and 0.9 gives the best results [Gre86].

Two different types of crossover are used in the experiments: *One-point crossover* and *Uniform crossover*. These types of crossover work as follows (using bit strings):

One-point crossover

Take two bit strings as parents. Choose a *crossover point* (a random point somewhere between the first and the last bit) and exchange the parts of the two parents after this crossover point. This way, two children are created, as shown in the next example.

```
parent 1:  000|00000       child 1:  00011111
parent 2:  111|11111       child 2:  11100000
              |
       crossover point
```

Uniform crossover

Take two bit strings as parents and create two children as follows: for each bit position on the two children, decide randomly which parent contributes its bit value to which child. Next, an example is shown:

```
parent 1:  00000000        child 1:  00101101
parent 2:  11111111        child 2:  11010010
```

Mutation

Start with the first genotype of the new population. Successively flip each bit with a certain probability p_m (called the *mutation rate*). Repeat this procedure for every next genotype in the population. In practice, the mutation rate p_m will be very small, in the order of magnitude of 0.01 to 0.001 for example. So, with bit strings, 1 bit in every 100 or 1000 bits will actually be flipped.

The mutation operator plays an important role in maintaining some *diversity* in the population. Crossover alone is not able to introduce a new value at a certain bit position when all the individuals in the population have the same value at this bit position. So, the task of mutation is primarily to prevent the population to converge to one specific point in the landscape, and to maintain some evolvability.

Now that the three operators are applied, the fitness of each genotype in the new population is determined, and the operators are applied again. This process is repeated for a fixed number of generations.

Schema processing

The notion of a *schema* is central to understand how a GA works. Schemata are sets of individuals in the search space, and the GA is thought to work by directing the search towards schemata containing highly fit regions of the search space (i.e. hilltops in the fitness landscape).

If the GA uses bit strings of length L as genotypes, then a schema is defined as an element of $\{0, 1, *\}^L$. So, a schema looks something like `1**01*00*`, where a `*` means *don't care*, either value (`0` or `1`) is allowed. A bit string b that matches the pattern of a schema s is said to be an *instance* of s. Fot example, both `00` and `10` are instances of `*0`.

In schemata, `0`'s and `1`'s are called *defined bits*. The *order* of a schema is the number of defined bits in that schema. The *defining length* of a schema is the distance between the first and the last defined bit. For example, the schema `1**01*00*` is of order 5 and has defining length 7.

The fitness of a schema is defined as the average of the fitness values of all bit strings that are an instance of this schema. For large string lengths, this is of course impossible to calculate for every schema, but the fitness of any bit string in the population gives some information about the fitness of the 2^L different schemata of which it is an instance. So, an explicit evaluation of a population of M individual strings is also an implicit evaluation of a much larger number of schemata.

The *building block hypothesis* ([Hol92, Gol89]) states that a GA works well when short, low-order, highly fit schemata (so-called *building blocks*) are recombined to form even more highly fit higher-order schemata. The ability to produce fitter and fitter partial solutions by combining building blocks is believed to be the primary source of the GA's search power.

According to the *Schema Theorem* ([Hol92, Gol89]), short, low-order, above average schemata receive exponentially increasing trials in subsequent generations. Above average means a fitness above the average fitness of the current population. So, early on in the search the GA *explores* the search space by processing as many different schemata as possible, and later on it *exploits* biases that it finds by converging to instances of the most fit schemata it has detected.

This strong convergence property of the GA is both a strength and a weakness. On the one hand, the fact that the GA can identify the fittest parts of the space very quickly is a powerful property. On the other hand, since the GA always operates on finite size populations, there is inherently some sampling error in the search, and in some cases the GA can magnify a small sampling error, causing *premature convergence.*

Another problem with GA's is *crossover disruption.* The building block hypothesis states that building blocks must be combined to ever fitter and longer schemata. But from the mathematical formula that supports the Schema Theorem ([Hol92, Gol89]), it follows that longer, higher-order schemata are more sensitive to being disrupted by crossover than shorter, low-order ones. So, crossover should on the one hand combine building blocks to longer, highly fit schemata, but on the other hand avoid disrupting them again as much as possible.

Both problems, premature convergence and crossover disruption, are examined in later chapters (see Chapters 5 and 6). To do this, two different GA's are applied to the fitness landscapes. Both GA's have the same parameter values, but the first one uses one-point crossover (GA-ONEP), while the second one uses uniform crossover (GA-UNIF). The implementation of the two GA's is as follows:

GA-ONEP

Population size:	50
Selection:	Deterministic tournament selection, s=3
Crossover:	One-point crossover, p_c=0.75
Mutation:	p_m=0.005

GA-UNIF

Population size:	50
Selection:	Deterministic tournament selection, s=3
Crossover:	Uniform crossover, p_c=0.75
Mutation:	p_m=0.005

3.1.4 Hybrid Genetic Algorithm

A Hybrid Genetic Algorithm (HGA) is a combination of a Genetic Algorithm (a global search strategy) with a local search strategy (see [Dav91]). It comes down to applying the local search strategy to a population of genotypes, then applying one generation of the GA, then the local search strategy again, etc. A GA can for example be combined with hillclimbing. First, let all the members of the population climb to a nearby hilltop. Next, apply crossover (and possibly mutation) to this population of local optima. Repeat this cycle for a number of generations. The idea behind this is that the locations of local optima may give some information about the locations of other, hopefully better, local optima.

Here, a GA combined with random ascent hillclimbing with memory (see section 3.1.1) is used. In the GA, an integrated selection recombination operator is used, called *Elitist Recombination*. This operator works as follows [TG94]:

Elitist Recombination

1. Random shuffle the population
2. For every mating pair
   (a) Generate offspring
   (b) Keep the best two of each family
   (= 2 parents + 2 offsprings)

So, with this operator, children are only allowed to enter the population if they are fitter than (one of) their parents. In the implementation that is used here, the offspring is generated with one-point crossover, with a crossover rate p_c of 1.0, so crossover is always applied. The exact implementation of the Hybrid Genetic Algorithm is as follows:

Hybrid Genetic Algorithm (HGA)

1. Initialize a random population of bit strings.
2. Apply Elitist Recombination to the population using one-point crossover (c=1.0)
3. Apply random ascent hillclimbing with memory to every member of the population
4. Repeat steps 2 and 3 for a preset number of function evaluations

A population size of 10 is taken for all experiments.

3.2 Performance measures

To compare the different search strategies that are introduced in the last section, they are evaluated by some performance measures. These measures are set out against the number of function evaluations done by a search strategy.

First of all, the maximum fitness in the population is monitored. For hillclimbing, which does not use a population, the best fitness found so far is taken. However, this performance measure only gives a snapshot at a certain time during the search. Therefore, the on-line and off-line performance (see [Gol89]) are measured as well. These measures keep track of all the function evaluations done by a strategy throughout the search.

As explained in Section 3.1.3, premature convergence of a population can be a problem for a Genetic Algorithm. So, it would be of interest to monitor the diversity of a population during a search. The mean Hamming distance is such a measure of population diversity, and it will be used here too.

In the following subsections, the on-line and off-line performance, as well as the mean Hamming distance, are explained in more detail.

3.2.1 On-line performance

The *on-line* (ongoing) *performance* is an average of all function evaluations done by a search strategy up to and including the current evaluation T. The on-line performance $onl_s(T)$ of strategy s is defined as:

$$onl_s(T) = \frac{1}{T}\sum_{1}^{T} f(t)$$

where $f(t)$ is the fitness value on evaluation t. Generally speaking, if the on-line performance of a search strategy stays very low during a search, then the strategy is wasting too much evaluations on "bad" solutions.

3.2.2 Off-line performance

The *off-line* (convergence) *performance* is a running average of the best fitness values found by a search strategy up to a particular evaluation. The off-line performance $\mathit{offl}_s(T)$ of strategy s is defined as:

$$\mathit{offl}_s(T) = \frac{1}{T}\sum_{1}^{T} f^*(t)$$

where $f^*(t)$ is the best fitness value encountered up to evaluation t. The off-line performance is a measure of how quickly the search strategy finds the optimal value (or "converges" to the optimum).

If, for example, at time $T = 5$ five genotypes have been evaluated by strategy s, with fitness 10, 8, 20, 2 and 15 respectively, then the on-line performance $\mathit{onl}_s(5)$ is $\frac{10+8+20+2+15}{5} = 11$, and the off-line performance $\mathit{offl}_s(5)$ is $\frac{10+10+20+20+20}{5} = 16$.

3.2.3 Mean Hamming distance

The *mean Hamming distance* (MHD) is a measure of population diversity. It is defined as the average value of the Hamming distances (see Section 2.2.1) between every two individuals of a population of bit strings:

$$MHD = \frac{1}{0.5n(n-1)}\sum_{i \neq j} HD(i,j)$$

where n is the population size, i and j range over (different) individuals of the current population, and $HD(i,j)$ is their Hamming distance.

Here, a normalized MHD is used, by taking the normalized Hamming distance between two bit strings (that is, the Hamming distance divided by the length of the bit strings, see Section 2.2.1). In this way, the MHD is independent of the length of the bit strings, and is a number between 0 and 1. A MHD of about 0.5, then, means that about half the bits of two arbitrary bit strings in the population differ in their value. This will be the case when a population of bit strings is created at random. A MHD of 0 indicates that all bit strings in the population are equal, so the population has completely converged onto one specific point in the fitness landscape.

3.3 Summary

In this chapter several search strategies are introduced that are applied to different fitness landscapes. These strategies include two types of hillclimbing (RAHCM and RNAHC), Long jumps, two Genetic Algorithms (GA-ONEP and GA-UNIF), and a Hybrid Genetic Algorithm (HGA). All these strategies apply one or more genetic operators to the genotypes they use during the search. Some of these operators are biologically inspired, others are purely artificial. By comparing the performance of the different strategies, hopefully a better understanding can be gained in the way these operators work or how useful they are. This can contribute to a better understanding of both problem solving and the process of evolution.

The performance of the search strategies is monitored by different performance measures. The first, and most important one, is the maximum fitness. For hillclimbing, the maximum fitness found up to a certain time is monitored, while for the other search strategies the maximum fitness in the population is recorded. This seems not quite fair, but since hillclimbing is usually considered as a benchmark, or "minimal performance", that works quite well on a multitude of problems, the performance of other strategies can be compared with the maximum found by hillclimbing. Besides, in this thesis the interest is focused on the population flow in general: not only how fast an optimum is found and how good this optimum is, but also how this optimum was reached and whether this optimum can be maintained in a population for a longer period of time.

Other performance measures are the on-line and off-line performance, which keep track of all the function evaluations done by a search strategy throughout a search. In case of a population-based strategy, the mean Hamming distance is also monitored, which gives a measure of the diversity of a population during a search.

To be able to relate the performance of a search strategy to the structure of the underlying fitness landscape, this structure has to be known first. Therefore, the next chapter proposes a way to express and determine the global structure of a fitness landscape.

Chapter 4

The Structure of Fitness Landscapes

To find a theory that relates the structure of a fitness landscape to the flow of a population on it, it is desirable to have some mathematical expression for this structure. But as already mentioned in Section 2.2.4, the structure of a fitness landscape incorporates many things, like its dimensionality, the number and average height of local optima, etc. One way to mathematically express the global structure of a fitness landscape, however, is by its correlation structure.

The *correlation structure* of a fitness landscape is determined by the fitness differentials between neighboring points in the landscape. Small differences in fitness between neighboring points gives a highly correlated landscape, large differences in fitness gives an uncorrelated landscape, with a whole range of more or less correlated landscapes in between. From this correlation structure, a *correlation length* can be derived, which somehow denotes the largest "distance" between points at which the fitness of one point still provides some information about the expected value of the fitness of the other point.

Different procedures have been used to measure and express the correlation structure and correlation length ([Wei90, MWS91, Lip91]). This chapter first proposes a more complete procedure based on the one introduced in [Wei90], and on a time series analysis known as the Box-Jenkins approach. Next, the results of applying this procedure to different NK-landscapes (see Section 2.3) are presented. Finally, some conclusions are drawn from these results.

4.1 The correlation structure

4.1.1 Measuring correlation

Weinberger introduced a procedure to measure the correlation structure of fitness landscapes [Wei90]. The idea is to generate a random walk on the landscape via neighboring points. In case of bit strings as genotypes and the Hamming distance as metric (see Section 2.2.1), this means that at every step one randomly chosen bit in the string is flipped, so-called *point mutation*. At each step the fitness of the genotype encountered is recorded. This way, a time series of fitness values is generated. Next, the autocorrelation function is used to determine the correlation structure of this time series.

The *autocorrelation function* ρ_i relates the fitness of two genotypes along the walk which are i steps (called *times lags* in case of a time series) apart. The autocorrelation for time lag i of a time series $y_t, t = 1, .., T$ is defined as:

$$\rho_i = Corr(y_t, y_{t+i}) = \frac{E[y_t y_{t+i}] - E[y_t]E[y_{t+i}]}{Var(y_t)}$$

where $E[y_t]$ is the expected value of y_t and $Var(y_t)$ is the variance of y_t. It always holds that $-1 \leq \rho_i \leq 1$. If $|\rho_i|$ is close to one, then there is much correlation between two values i steps apart; if it is close to zero, then there is hardly any correlation. Estimates of these autocorrelations are:

$$r_i = \frac{\sum_{t=1}^{T-i}(y_t - \overline{y})(y_{t+i} - \overline{y})}{\sum_{t=1}^{T}(y_t - \overline{y})^2}$$

where $\overline{y} = \frac{1}{T}\sum_{t=1}^{T} y_t$ and $T \gg 0$.

An important assumption made here is that the fitness landscape is *statistically isotropic*. This means that the statistics of the time series generated by a random walk via neighboring points are the same, regardless of the starting point. The significance of statistical isotropy is that the random walk is "representative" for the entire landscape, and thus that the correlation structure of the time series can be regarded as the correlation structure of the landscape.

There still seems to be no agreement about an exact definition of the *correlation length* of a fitness landscape, since everybody uses its own definition ([Wei90, MWS91, Lip91]). The correlation length gives an indication of the largest "distance", or time lag, between two points at which the fitness of one point still provides some information about the expected value of the fitness of the other point. In other words, the correlation length τ is the largest time lag i for which there still exists some correlation between two points i steps apart.

In statistics, it is usual to compare an estimated value with its two-standard-error[1] bound, to see whether the estimated value is significantly different from zero. For the r_i, the estimates of the ρ_i, this two-standard-error bound is $\pm 2/\sqrt{T}$ (see [J$^+$88]). So, it is proposed here to take as correlation length τ one less than the first time lag i for which the estimated autocorrelation r_i falls inside the region $(-2/\sqrt{T}, +2/\sqrt{T})$, and thus becomes (statistically) equal to zero. This way, the correlation length τ is the largest time lag i for which the correlation between two points i steps apart is still statistically significant.

[1] The statistical estimation of some variable will never be exact, but contains some uncertainty. The standard error of an estimated value gives an indication of the amount of this uncertainty.

4.1.2 Time series analysis

The procedure introduced in [Wei90] only calculates the autocorrelations from the obtained time series, and derives a correlation length from them. It is proposed here to expand this correlation analysis to a more complete time series analysis. This involves identifying an appropriate model that adequately represents the data generating process (in this case the random walk), estimating the parameters of this model, and applying statistical tests to see how well the estimated model approximates the given data and what the explanatory and predictive value of the model is.

In other words, a model of the form

$$y_{t+1} = f(y_t, y_{t-1}, \cdots, y_0)$$

is derived from the observed data, that can be used to simulate the outcome of a random walk on the landscape or to predict future values in a time series generated by such a walk. Different landscapes can then be compared in terms of these models, which are used to express the correlation structure of these landscapes.

Section 4.2 introduces such a complete time series analysis that is based on the estimation of the autocorrelations of a given time series. This time series analysis is known as the Box-Jenkins approach, and it will be applied here to time series that are generated by a random walk on NK-landscapes.

4.1.3 Handling other operators

Genetic operators that give rise to steps of a distance larger than one (in terms of the metric that is used to define the fitness landscape) will experience another correlation structure. In terms of bit strings, operators other than point mutation will experience other correlation structures on a landscape defined by Hamming distance. One step of an arbitrary operator other than mutation (for example long jumps or crossover) will, in general, end up in a point that has a Hamming distance of more than one from the point that was started from. The correlation between two points "one step apart" will thus be different for each operator. So, each operator experiences the landscape in a different way.

Compare this with a mountaineers-club which has a couple of camps in the Alps. Consider a mountaineer walking along a route passing these camps. The only things this mountaineer can observe at every step he takes, are his coordinates and his altitude. From these observations he has to construct a picture of the landscape he is walking through. Now a fellow mountaineer is hopping in a helicopter from one camp to another. He is also able to record only his coordinates and altitude at every camp he lands. The picture he constructs from his data will be different from the picture his walking fellow made earlier on, while both men travelled through the same landscape!

A procedure to deal with different operators is due to Manderick et al. [MWS91]. This procedure involves generating a random population of genotypes (the parent population), applying the operator of interest to this population, thus creating an offspring population, and then calculating the correlation coefficient between the fitness values of the parents and the offspring. This procedure is done for the first generation. If the correlation coefficient for more generations is wanted, then this procedure can be repeated, this time with the offspring population acting as parent population. Because no new genotypes are introduced during this process, there is a chance that the population converges to some extent, and this will be reflected in the calculations of the correlation coefficient, causing a bias in the outcome.

Therefore, it is proposed here to use the procedure introduced in Section 4.1.2 for other operators as well. Instead of walking along neighboring points in the landscape (that is, using point mutation in case of bit string and Hamming distance), a random walk can be generated by the operator of interest[2]. The complete time series analysis applied to the time series generated in this way, gives insight into how this particular operator experiences the correlation structure of the landscape.

In fact, calculating the autocorrelationfor the first time lag of such a time series is equal to calculating the correlation coefficient for parent and offspring populations. Instead of creating an offspring population and calculating the correlation between the fitness of parents and offspring, the correlation between the original time series and the same series one time lag shifted is calculated (remember that the genotype encountered at time t is the parent of the genotype encountered at time $t+1$). Hence the term *auto*correlation, or correlation with oneself. In the same way, the correlations for larger time lags can also be calculated with only one time series, instead of a number of populations, thus avoiding the danger of a biased outcome due to convergence.

The next section introduces the Box-Jenkins approach, which is used to perform the complete time series analysis based on the estimated autocorrelations of the time series.

4.2 The Box-Jenkins approach

The Box-Jenkins approach [BJ70] is a very useful standard statistical method of model building, based on the analysis of a time series $y_1, y_2, \ldots, y_T$, generated by a stochastic process. The purpose of the Box-Jenkins approach is to find an ARMA model that adequately represents this data generating process. Once an adequate model is found, it can be used to make forecasts about future values, or to simulate a similar process as the one that generated the original data.

[2] Note that this gives a slight problem for binary operators (i.e. operators that use two parents instead of one). One way to overcome this problem is by choosing a second parent at random out of all possible genotypes and taking one of the two children, both with an equal chance, to proceed with.

An ARMA model represents an *autoregressive moving-average process*, and is obtained by combining an autoregressive (AR) process and a moving-average (MA) process. An AR process of order p (AR(p)) has the form

$$y_t = \alpha_1 y_{t-1} + \cdots + \alpha_p y_{t-p} + \varepsilon_t$$

where the stochastic variable ε_t is white noise, that is, $E[\varepsilon_t] = 0$, $Var(\varepsilon_t) < \infty$ for all t, and $Cov(\varepsilon_s, \varepsilon_t) = 0$ for $s \neq t$, so all ε_t are independent of each other. So, each value in an AR(p) process depends on p past values and some stochastic variable ε_t. An MA process of order q (MA(q)) has the form

$$y_t = \varepsilon_t + \beta_1 \varepsilon_{t-1} + \cdots + \beta_q \varepsilon_{t-q}$$

where ε_t is again white noise. So, each value in an MA(q) process is a weighted sum of members of a white noise series. An ARMA(p, q) process, then, is a combination of an AR(p) and an MA(q) process:

$$y_t = \alpha_1 y_{t-1} + \cdots + \alpha_p y_{t-p} + \varepsilon_t + \beta_1 \varepsilon_{t-1} + \cdots + \beta_q \varepsilon_{t-q}$$

The mean of a time series generated by one of these three processes is zero. If this is not wanted, then a constant c can be added to the model, resulting in a non-zero mean of the time series.

In economics (and business) the Box-Jenkins approach is used frequently when a model is needed to make forecasts about future values of some (partly) stochastic variable, for example the price of some commodity, or the index of industrial production. The approach consists of three stages:

1. *Identification*, in which a choice is made for one or more appropriate models, by looking at the (partial) autocorrelations of the time series;

2. *Estimation*, in which the parameters of the chosen model are estimated;

3. *Diagnostic checking*, which involves applying various tests to see if the chosen model is really adequate.

The three stages of the approach are explained in more detail below.

Identification

At the identification stage an appropriate model is specified on the basis of the correlogram and the partial correlogram. The correlogram of a time series y_t is a plot of the (estimated) autocorrelations (the r_i as given in Section 4.1.1) of this series against the time lag i. The partial correlogram is the plot of the (estimated) partial autocorrelations of the time series against the time lag. It will not be explained here how to calculate the partial autocorrelations (see [BJ70, Gra89, J$^+$88]), but the i'th partial autocorrelation can be interpreted as the estimated correlation between y_t and y_{t+i}, after the effects of all intermediate y's on this correlation are taken out. The choice of model can now be made on the following basis:

- If the correlogram tapers off to zero and the partial correlogram suddenly "cuts off" after some point, say p, then an appropriate model is AR(p). To determine this cut-off point p, the partial autocorrelations are compared with a two-standard-error bound, which is $2/\sqrt{T}$ (T being the length of the time series).
- If the correlogram "cuts off" after some point, say q, and the partial correlogram tapers off to zero, then an appropriate model is MA(q). Here the "cut-off" point is also determined by comparing the autocorrelations with their two-standard-error bound of $2/\sqrt{T}$.
- If neither diagram "cuts off" at some point, but both taper off, then an appropriate model is ARMA(p,q). The values of p and q have to be inferred from the particular pattern of the two diagrams.

Estimation
Once the appropriate model is chosen, the parameters of this model can be estimated. This is achieved by using the estimates of the autocorrelations. From these values, estimates for the parameters of the model can be derived (see [BJ70, Gra89, J$^+$88]).

As a measure of significance of the estimated parameters the *t-statistic* is used. This statistic is defined as the estimated value of the parameter divided by its estimated standard error. Because the estimation of a parameter will never be exact, an interval of two times the standard error on both sides of the estimate determines a so called *95% confidence interval*. The probability that the real value of the parameter will fall inside this interval is 95%. But if zero also falls inside this interval, then the parameter could just as well be equal to zero. For this reason, a parameter is called *significant* (meaning significantly different from zero), if the absolute value of the t-statistic of its estimate is greater than two, because zero will then be outside the 95% confidence interval.

As a measure of "goodness of fit" of the estimated model, the R^2 is used. This value is a measure of the proportion of the total variance in the data accounted for by the influence of the explanatory variables of the estimated model. A value of R^2 close to one means that the explanatory variables can explain the observed data very well. A value of R^2 close to zero means that the stochastic component of the model plays a dominant role (or it could be that there exist more explanatory variables than there are currently in the model; this will not be the case here, because it is assumed that in the identification stage an appropriate model is already chosen).

Diagnostic checking
Before the estimated model is used, it is important to check that it is a satisfactory one. The usual test is to fit the model on the data and calculate the autocorrelations of the residuals (the difference between the observed values and those predicted by the estimated model). These residuals should be white noise, so all the autocorrelations should not be significantly different from zero. To check this, the residual autocorrelations are compared

with a two-standard-error bound (also $2/\sqrt{T}$). Another test is to fit a slightly higher-order model and then to see whether the extra parameters are significantly different from zero. So, if an AR(p) model is estimated, an AR($p+1$) model could also be estimated and the significance of the extra parameter should be checked (it should be insignificant).

Summarizing, the Box-Jenkins approach is used to find an appropriate model for a given time series generated by some stochastic process, to make forecasts for, or simulations of the process that generated the original data. In the next section, the results of applying the Box-Jenkins approach to time series of fitness values generated by random walks on NK-landscapes are presented.

4.3 The correlation structure of NK-Landscapes

To determine the correlation structure of NK-landscapes, the Box-Jenkins approach is applied to time series of fitness values, generated by random walks on these landscapes. Since NK-landscapes are defined by bit strings as genotypes and the Hamming distance as metric (see Section 2.3), at least the operator point mutation is used to generate random walks. This operator visits neighboring points in the NK-landscape, so the results obtained for this operator can be regarded as *the* correlation structure of the NK-landscape.

Furthermore, because all search strategies introduced in Chapter 3 use at least one of the operators mutation, long jumps or crossover, these last two operators are also used to generate random walks (only one-point crossover is used here; uniform crossover is not considered). The results obtained for these operators indicate how they experience the correlation structure of the NK-landscape. The three types of random walks are implemented as follows:

Point mutation: At every step one bit, which is selected at random, is flipped.

One-point crossover: At every step a second parent is randomly chosen out of all possible bit strings. A crossover point is selected at random, and the parts of the parents after this crossover point are exchanged, creating two children. One of these two children is selected for the next step (each with chance 0.5).

Long jumps: At every step each bit in the string is flipped with chance 0.5.

The length of the random walks is 10,000 steps, and the autocorrelations for the first 50 time lags are estimated. NK-landscapes with the following values for N and K are considered: N=100, K=0, 2, 5, 25, 50 and 99. Both random and nearest neighbor interactions are considered. The Box-Jenkins approach is carried out with the statistical package TSP (Time Series Processor). The following subsections present the results for the different operators.

4.3.1 Results for point mutation

Identification

The correlograms for the different values of K are given in Figures 4.1 (random interactions) and 4.2 (nearest neighbor interactions), together with the two-standard-error bound of $2/\sqrt{T}$, or 0.02 for T=10,000. The correlograms all taper off to zero (except for K=99; here the autocorrelations almost immediately drop to zero), so an AR(p) or an ARMA(p,q) process should be most appropriate here. The graphs show clearly that the correlation length decreases as K increases, and that there is not much difference between random interactions and nearest neighbor interactions. Table 4.1 gives the correlation lengths (as defined in Section 4.1.1) for the different values of K.

K	random interactions	nearest neighbor interactions
0	>50	>50
2	>50	>50
5	49	>50
25	18	14
50	5	7
99	2	3

Table 4.1: The correlation lengths for the operator point mutation on NK-landscapes for N=100 and different values of K.

What is striking here is the fact that for K=99 there still is some correlation left. According to Kauffman [Kau93] it should be expected that the correlation is zero in the case of $K = N - 1$, because the landscape is completely random. So the time series generated by a random walk should be white noise (i.e. completely random) around the mean of the series. But although the estimated correlations are very small, as the graphs show, statistically they are significantly different from zero. Repeating the procedure for this value of K always gives the same sort of result, so it is not "just an accident".

A possible explanation for these minor correlations is that for $K = N - 1$ the fitnesses of the local optima become very small due to the large amount of conflicting constraints. All fitness values vary just slightly around the mean of 0.5. The fact that the fitnesses of different points do not differ very much, although they are completely random, might introduce some slight correlations, which, strictly speaking, are not actually present.

To see which of the two models (AR(p) or ARMA(p,q)) is the most appropriate, the partial correlogram for K=0 (random interactions) is shown in Figure 4.3. This plot shows that the first partial autocorrelation is almost equal to one, and thus well outside the two-standard-error bound of 0.02. The other partial autocorrelations are all within this bound (apart from one minor exception). So, it is clear that the partial correlogram "cuts off" after one time lag, and thus an AR(1) model is the right choice here.

The partial correlograms for the other values of K and for nearest neighbor interactions look very similar, except for K=99. In this case, the first two or three partial autocorrelations are outside the two-standard-error bound, indicating an AR(2) or AR(3) process. Although these partial autocorrelations are very small, they are statistically significant. This is also an ever-recurring result when repeating the whole procedure.

Estimation
In all cases an AR(1) process of the form

$$y_t = c + \alpha_1 y_{t-1} + \varepsilon_t$$

is estimated. The constant is added because the mean of the time series is not equal to zero. Table 4.2 shows the results of the estimation. The t-statistics (see Section 4.2) of the estimated parameters are shown in parentheses.

	random interactions				nearest neighbor interactions			
K	c	α_1	$Var(\varepsilon_t)$	R^2	c	α_1	$Var(\varepsilon_t)$	R^2
0	0.00850 (9.59)	0.98181 (518.20)	0.0000182	0.964	0.00959 (9.94)	0.98090 (510.85)	0.0000177	0.963
2	0.01620 (13.08)	0.96629 (375.48)	0.0000467	0.934	0.01763 (13.47)	0.96431 (364.39)	0.0000461	0.930
5	0.03309 (18.46)	0.93400 (261.59)	0.0000964	0.872	0.02999 (17.55)	0.94005 (275.72)	0.0000961	0.884
25	0.12857 (38.35)	0.74272 (110.91)	0.0003779	0.552	0.13140 (38.86)	0.73687 (109.00)	0.0003808	0.543
50	0.25419 (58.29)	0.49141 (56.41)	0.0006357	0.242	0.25494 (58.40)	0.49003 (56.21)	0.0006349	0.240
99	0.48293 (96.51)	0.03374 (3.38)	0.0008257	0.001	0.49013 (97.78)	0.02063 (2.06)	0.0008335	0.000

Table 4.2: The results of the estimation of an AR(1) process for the operator point mutation on NK-landscapes for N=100 and different values of K.

The table shows that all parameters are significant (t-statistic > 2). It also shows clearly that the correlation coefficient (α_1) decreases and the variance of the error term ($Var(\varepsilon_t)$) increases as K increases. This explaines the fact that the correlation length decreases for increasing K. Note that the correlation coefficient decrease linearly with increasing K. Furthermore, the R^2 decreases as K increases, so the estimated model is less capable of explaining the observed data, apart from random variances, for higher values of K.

The table, just as the correlograms, also shows no difference between random and nearest neighbor interactions.

It appears that the estimated α_1 and the R^2 for K=99 are both very small. So, the estimated model for $K = N - 1$ is hardly any different from an ordinary white noise series around the mean of the series, which is, as said earlier, theoretically expected.

Diagnostic checking
Figure 4.4 shows the first 25 residual autocorrelations for K=0 (random interactions). This plot shows that they are all well within the two-standard-error bound of 0.02 (apart from one minor exception). The plots for the other values of K and for nearest neighbor interactions look very similar.

To make absolutely sure, an AR(2) model is estimated for all cases. The results are presented in Table 4.3.

	random interactions		nearest neighbor interactions	
K	α_2	t-statistic	α_2	t-statistic
0	0.00624	0.62	-0.00548	-0.55
2	-0.01104	-1.10	-0.01437	-1.44
5	-0.00716	-0.72	0.00642	0.64
25	0.00526	0.53	0.00955	0.96
50	0.00812	0.81	-0.00746	-0.75
99	0.03189	3.19	0.04821	4.83

Table 4.3: The results of the overestimation of the chosen model for different values of K.

As expected, the extra parameter is not significantly different from zero in all cases except for K=99. This exception is not surprising, because the partial autocorrelations already suggested an AR(2) or AR(3) process. But as noted above, the estimated parameters in this model are very small (as well as the R^2), and it effectively comes down to an ordinary white noise series. So, these two checks show that the chosen AR(1) model seems to be adequate in all cases, except for K=99.

A final remark here, is that it is important to note that the time lag i between two values in the time series is not the same as the Hamming distance between two genotypes i steps apart. Because a randomly chosen bit is flipped each step, the same bit can be flipped more than once in a sequence of steps. Generating a random walk of length 10,000 with point mutation as operator, and calculating the average (normalized) Hamming distance between two genotypes i steps apart for $0 \leq i \leq 50$, gives a result as shown in Figure 4.5. In this graph it can be seen that the Hamming distance increases less than linearly with the time lag i.

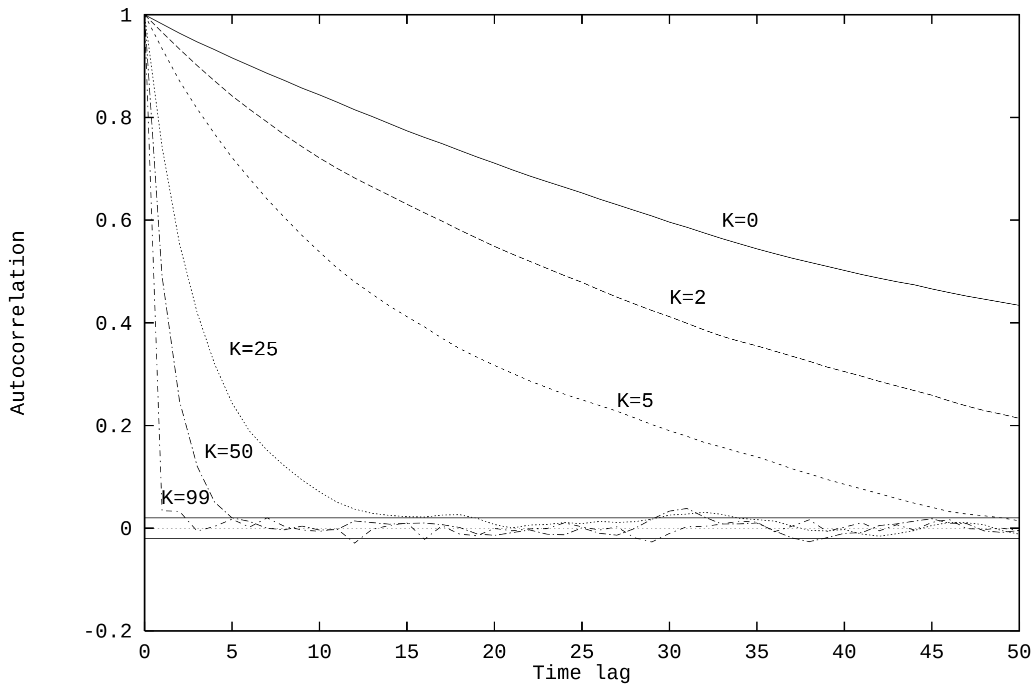

Figure 4.1: The first 50 autocorrelations for point mutation on NK-landscapes for N=100 and different values of K (random interactions).

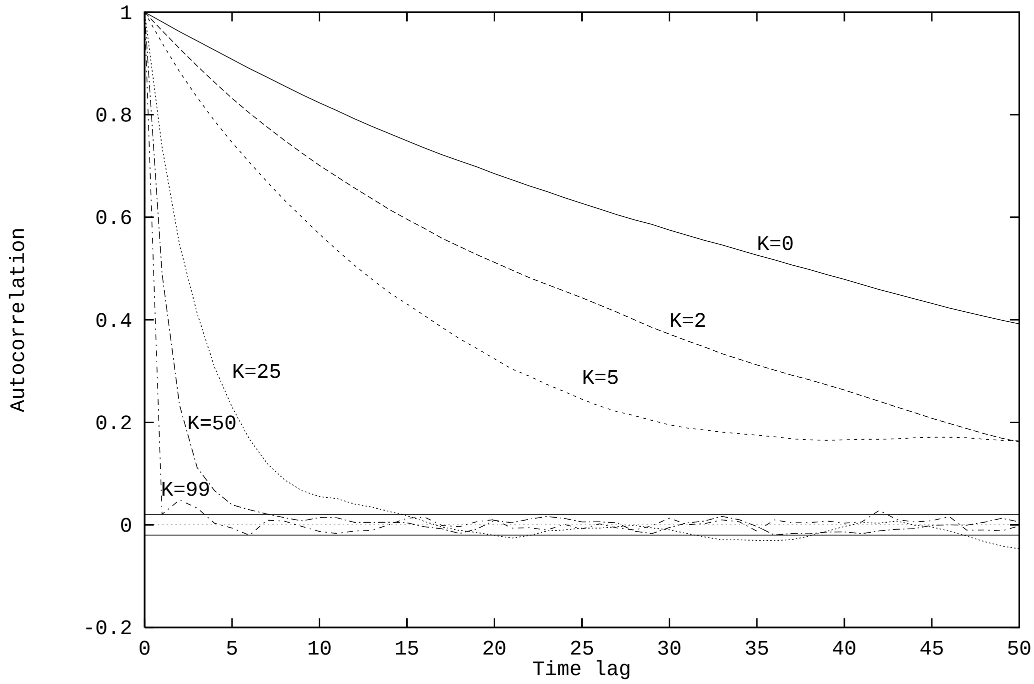

Figure 4.2: The first 50 autocorrelations for point mutation on NK-landscapes for N=100 and different values of K (nearest neighbor interactions).

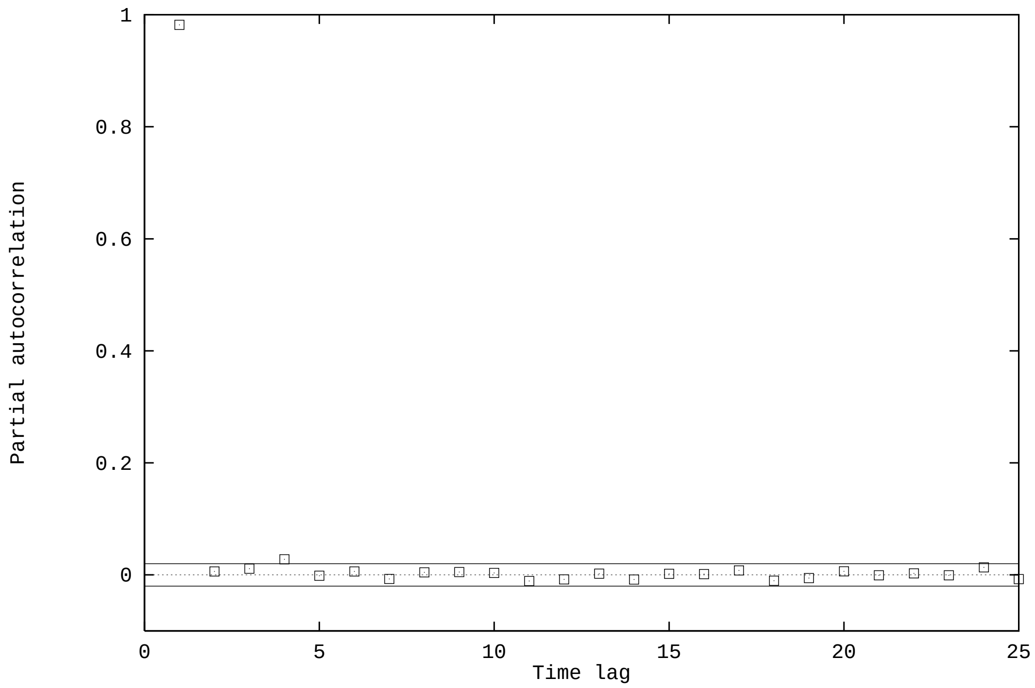

Figure 4.3: The first 25 partial autocorrelations for point mutation on NK-landscapes, N=100, K=0 (random interactions).

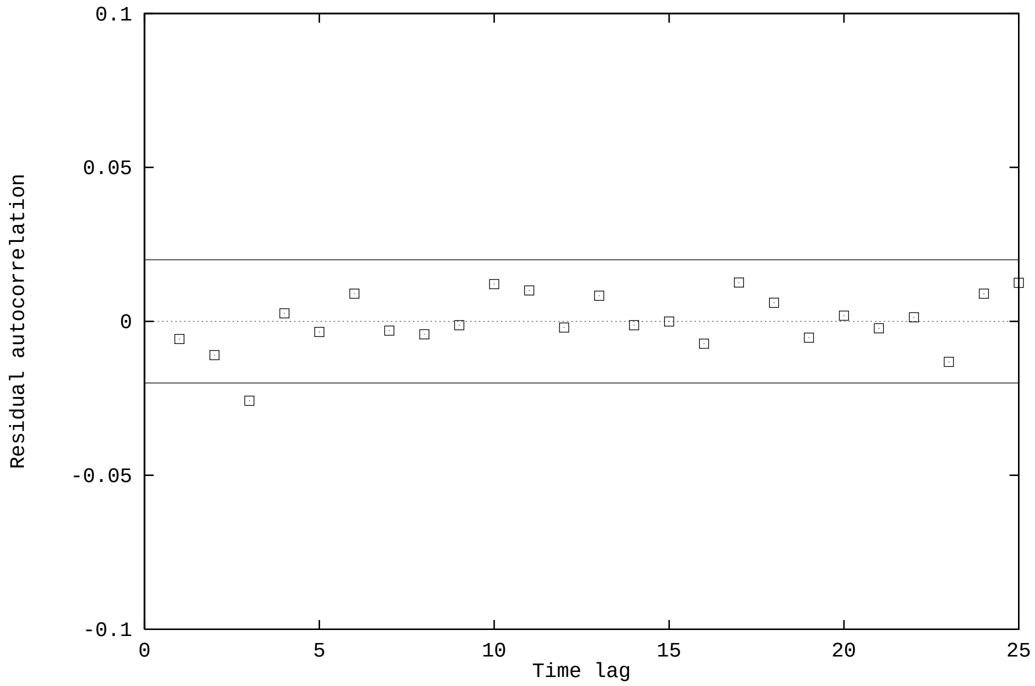

Figure 4.4: The first 25 residual autocorrelations for the estimated AR(1) model for point mutation on NK-landscapes N=100, K=0 (random interactions).

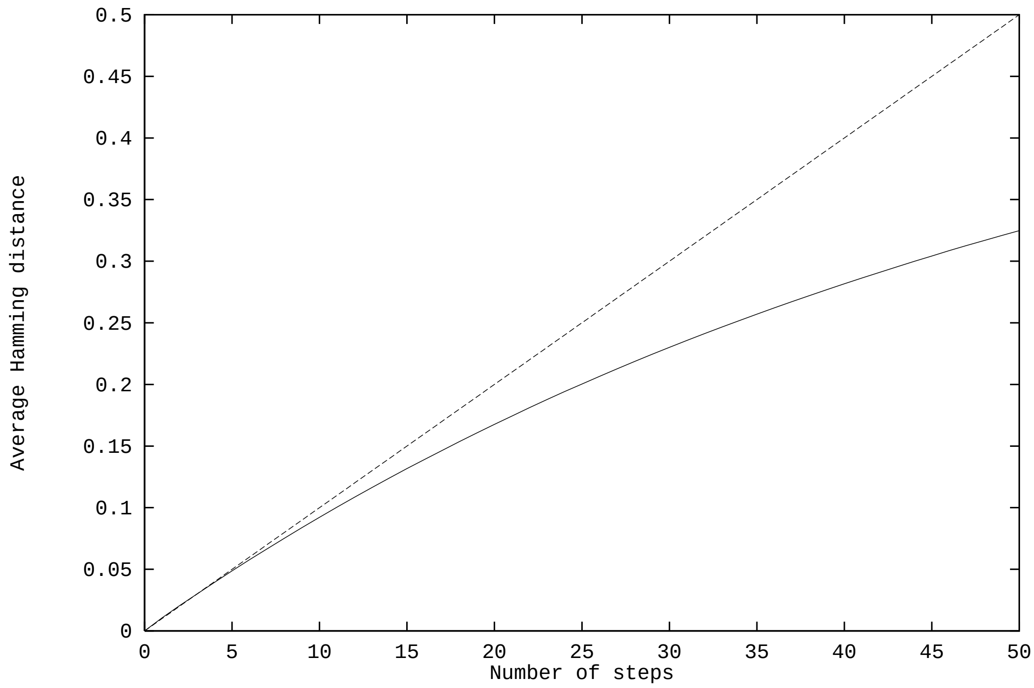

Figure 4.5: The average (normalized) Hamming distance of two genotypes i steps apart $(0 \le i \le 50)$ in a random walk generated with point mutation.

4.3.2 Results for crossover

Identification

The correlograms for crossover (that is, one-point crossover) for the different values of K are presented in Figures 4.6 (random interactions) and 4.7 (nearest neighbor interactions). These correlograms also taper off to zero, but much faster than for mutation. The graphs show furthermore that for crossover there is some difference between random interactions and nearest neighbor interactions. This is shown more clearly in Table 4.4, which shows the correlation lengths for the different values of K.

K	random interactions	nearest neighbor interactions
0	6	4
2	5	4
5	3	4
25	1	3
50	0	2
99	1	1

Table 4.4: The correlation lengths for the operator one-point crossover on NK-landscapes for N=100 and different values of K.

The table shows that the correlation dies out for random interactions, whereas for nearest neighbor interactions there still remains some correlation as K increases. This difference can be explained by the fact that the genetic operator one-point crossover makes use of *building blocks*, or co-adapted sets (see Section 3.1.3). If the epistatic interactions are spread randomly across the genotype, then every possible crossover point will affect the epistatic relations of almost all bits in the string, especially for larger values of K. But if the epistatic interactions are the K neighboring bits, then only the epistatic relations of the bits in the vicinity of the crossover point are affected (this is stated in another way in the *Schema Theorem*, which says that schemata of short defining length have a higher chance to survive under one-point crossover than longer ones; see Section 3.1.3). So, if more epistatic relations stay intact, more bits will keep the same fitness, and the fitness of the entire genotype in the next step will be more correlated to that of its parents.

Furthermore, it is again striking that both for random and nearest neighbor interactions there is some correlation for the case of $K = N - 1$, which is not expected. Repeating the procedure for this case a couple of times still gives the same result. The same explanation as for mutation can be given here, namely that the fitnesses of the local optima vary just a little around the mean of 0.5, introducing slight correlations, which are strictly speaking not present.

Looking at the partial correlogram for K=0 (random interactions), as shown in Figure 4.8, it can be concluded that an AR(1) process is the most appropriate model here too. The partial correlograms for the other values of K and for nearest neighbor interactions look very similar, except for K=50, random interactions. In this case, all partial autocorrelations are within the two-standard-error bound, indicating that a white noise series around the mean is the most appropriate model here. This is not surprising, because the correlation length is zero in this case, as Table 4.4 shows.

Estimation
As with mutation, an AR(1) process with constant is estimated here. The results are presented in Table 4.5 (t-statistics shown between parentheses).

	random interactions				nearest neighbor interactions			
K	c	α_1	$Var(\varepsilon_t)$	R^2	c	α_1	$Var(\varepsilon_t)$	R^2
0	0.26140	0.50278	0.0003423	0.253	0.23012	0.50130	0.0003581	0.251
	(57.52)	(58.21)			(57.57)	(57.93)		
2	0.31878	0.37191	0.0006060	0.138	0.26521	0.49274	0.0005581	0.243
	(67.56)	(40.06)			(58.21)	(56.62)		
5	0.37223	0.26104	0.0007311	0.068	0.26929	0.45817	0.0006116	0.210
	(76.42)	(27.04)			(60.85)	(51.54)		
25	0.47254	0.05400	0.0008388	0.003	0.35425	0.29229	0.0007491	0.085
	(94.56)	(5.41)			(73.87)	(30.56)		
50	0.49038	0.01925	0.0008214	0.000	0.42640	0.14640	0.0008310	0.021
	(97.92)	(1.92)			(86.14)	(14.80)		
99	0.48515	0.02921	0.0008209	0.001	0.48189	0.03048	0.0008354	0.001
	(96.95)	(2.92)			(96.84)	(3.05)		

Table 4.5: The results of the estimation of an AR(1) process for one-point crossover on NK-landscapes for N=100 and different values of K.

All parameters appear to be significant, except for the α_1 for K=50, random interactions (indicating again the absence of correlation in this case). The table also shows that the estimated value of α_1 is larger for nearest neighbor interactions than for random interactions for $0 < K < N-1$ (for the two extremes of K=0 and $K = N-1$ there is of course no difference between random and nearest neighbor interactions, see page 12). This indicates that there is indeed more correlation for one-point crossover between two points one step apart for in a landscape with nearest neighbor interactions, than in a landscape with random interactions. Also, the R^2 is larger for nearest neighbor interactions for $0 < K < N-1$, which means that the estimated model has more explanatory and predictive value than it does for random interactions. For crossover too, the model for K=99 is hardly distinguishable from ordinary white noise, considering the estimated value of α_1 and R^2.

Diagnostic checking

Figure 4.9 shows the first 25 residual autocorrelations for K=0 (random interactions). All are well within the two-standard-error bound. The plots for larger values of K and nearest neighbor interactions look very similar. Table 4.6 gives the results of the overestimation, the extra check on the adequacy of the chosen model. The table shows that all extra parameters are not significantly different from zero, indicating once more that the AR(1) model is adequate.

	random interactions		nearest neighbor interactions	
K	α_2	t-statistic	α_2	t-statistic
0	0.00412	0.41	-0.00623	-0.62
2	0.01767	1.77	-0.00866	-0.87
5	0.01198	1.20	0.00613	0.61
25	0.00013	0.01	0.01576	1.58
50	0.00386	0.39	0.01270	1.30
99	0.00284	0.28	0.00160	0.16

Table 4.6: The results of the overestimation of the chosen model for crossover for different values of K.

The average (normalized) Hamming distance between parent and child in a random walk generated with one-point crossover will be 0.25. This can be argued as follows: Because a second parent is chosen at random, on average half of the bits of the two parents will have a different value. The crossover point will be, also on average, somewhere halfway, so either the first half or the second half of the child will be equal to the first parent (remember that one of the two children is selected at random). The other half of the child will come from the second parent, so about half the bits in this part will be different from the first parent. As a result, about one quarter of the bits of the selected child will be different from the first parent (which was the one used for continuing the random walk one step ago).

The following simple example shows a situation where the Hamming distance between the two parents is 0.5, the crossover point is halfway the genotypes, and the Hamming distance between the child and the first parent is 0.25.

```
parent 1:   0000|0000
parent 2:   1010|1010
---------------------
child   :   0000|1010
```

Calculating the average Hamming distance between parent and child in a random walk of length 10,000 generated with crossover, confirms this result. Figure 4.10 gives the average Hamming distance of two genotypes i steps apart for $0 \leq i \leq 50$.

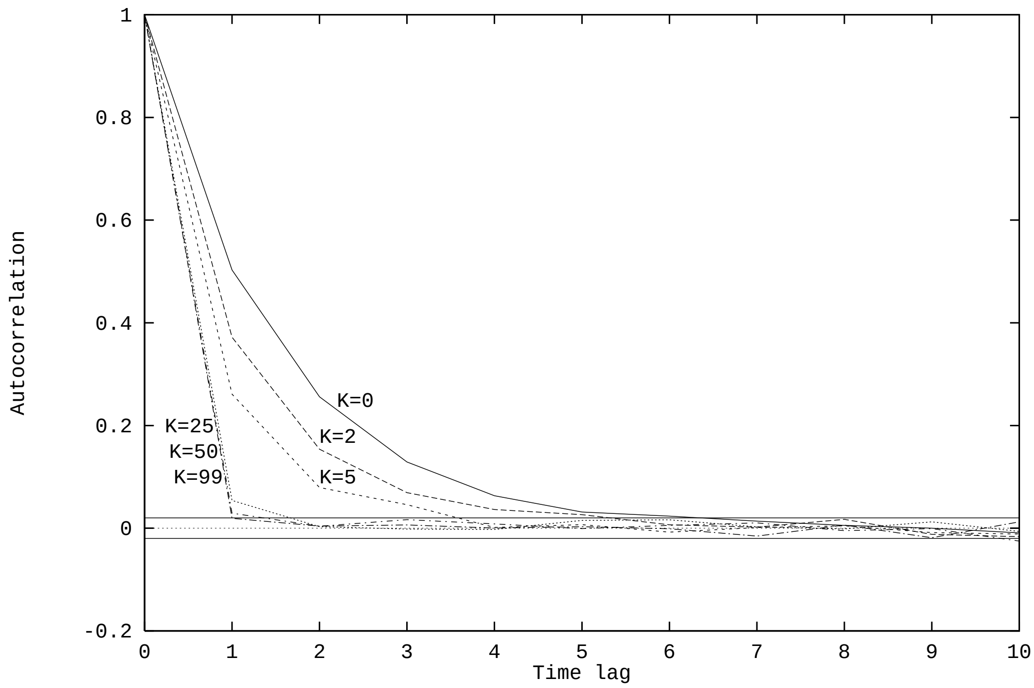

Figure 4.6: The first 10 autocorrelations for crossover on NK-landscapes for N=100 and different values of K (random interactions).

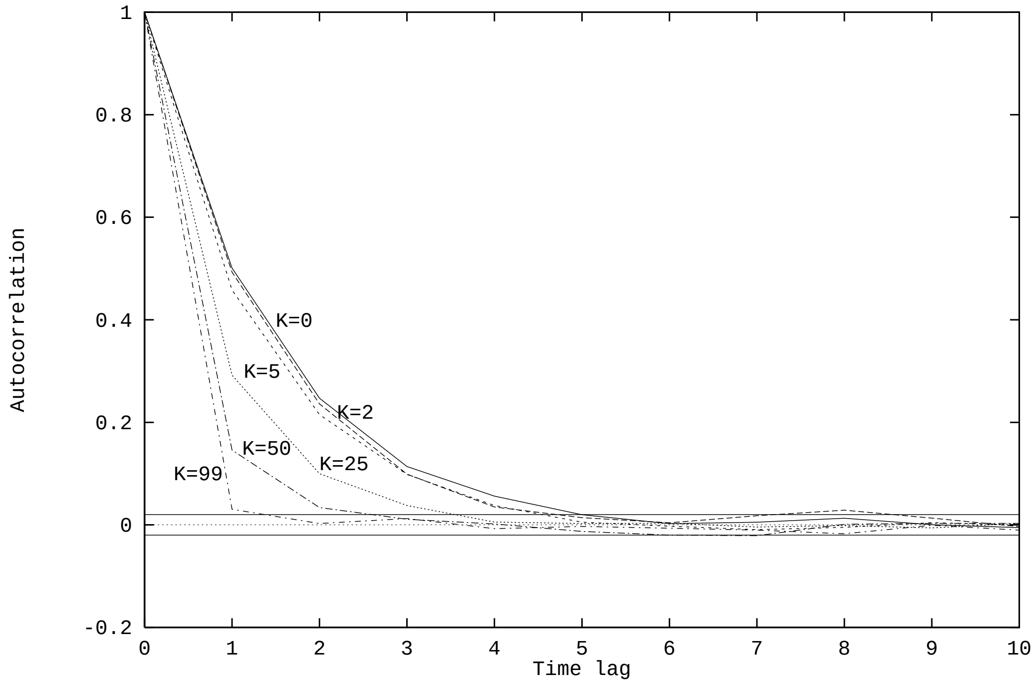

Figure 4.7: The first 10 autocorrelations for crossover on NK-landscapes for N=100 and different values of K (nearest neighbor interactions).

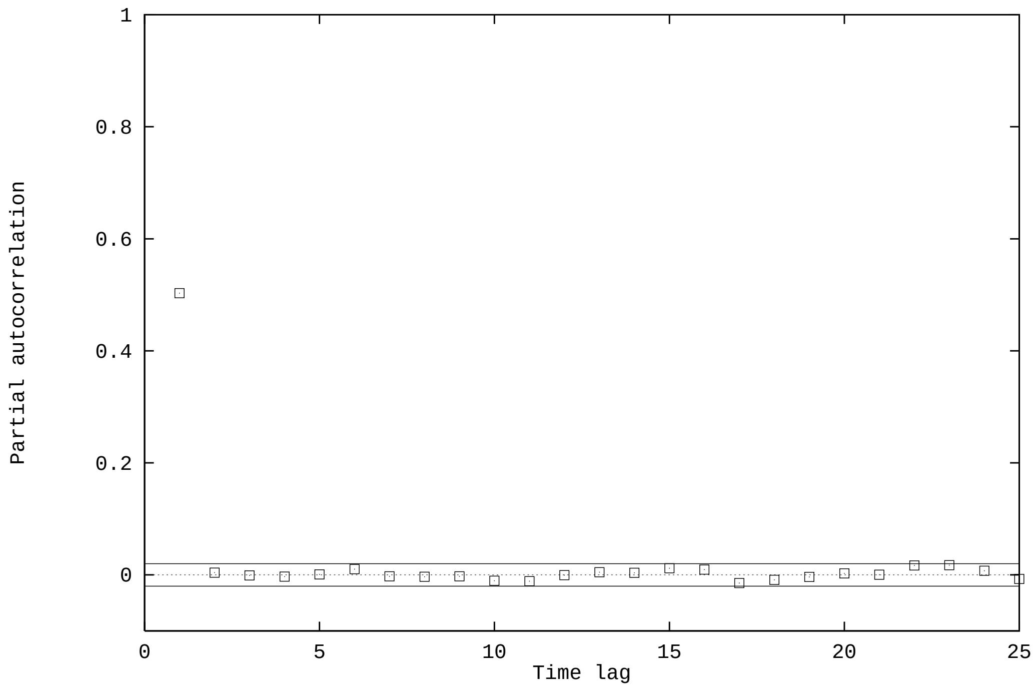

Figure 4.8: The first 25 partial autocorrelations for crossover on NK-landscapes, N=100, K=0 (random interactions).

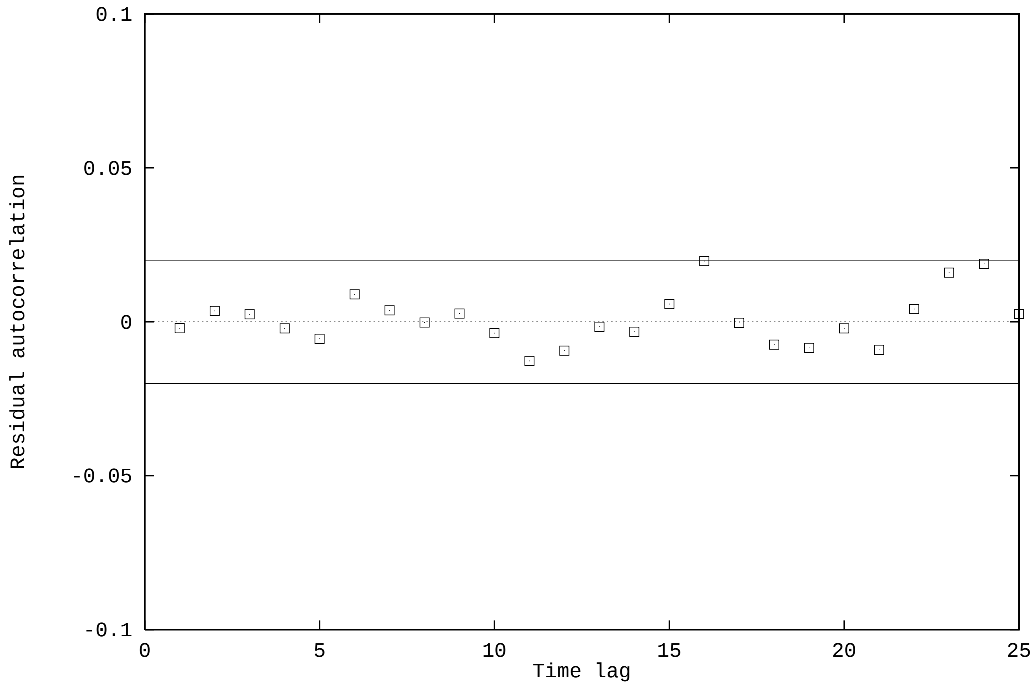

Figure 4.9: The first 25 residual autocorrelations for the estimated AR(1) model for crossover on NK-landscapes, N=100, K=0 (random interactions).

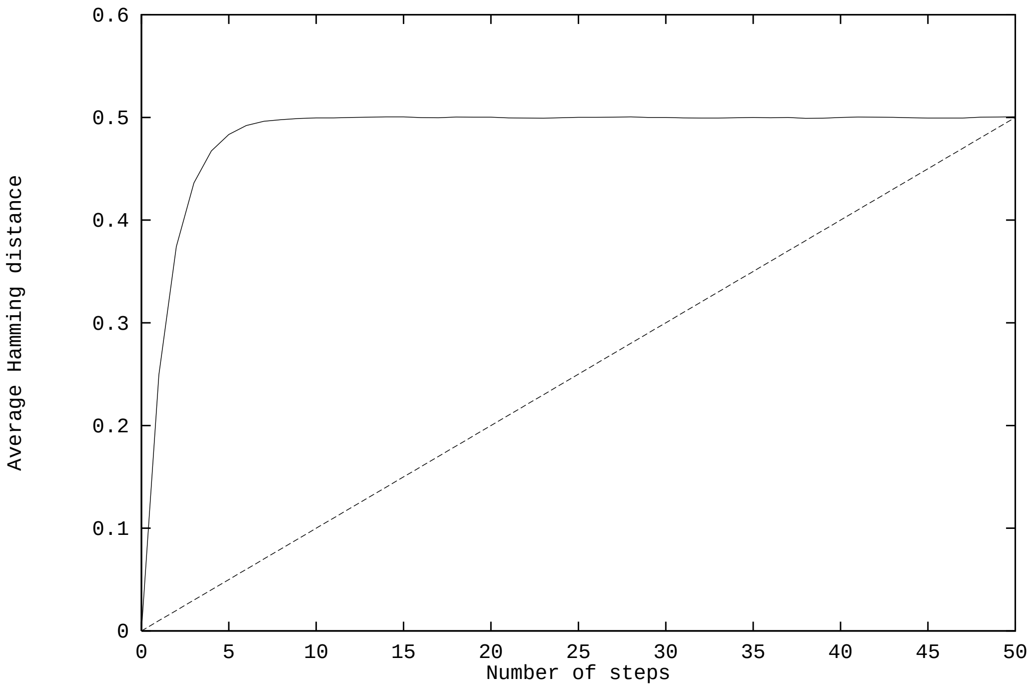

Figure 4.10: The average (normalized) Hamming distance of two genotypes i steps apart ($0 \leq i \leq 50$) in a random walk generated with one-point crossover.

4.3.3 Results for long jumps

For all values of K (both random and nearest neighbor interactions), the correlation length for long jumps turns out to be zero (results not shown). So a searcher jumping long distances in the landscape is encountering a totally random landscape, even for small values of K. The estimated parameters for the AR(1) process all proved to be insignificant (except for the constants of course), indicating that the only appropriate model is just a white noise series. So, for long jumps every landscape looks just the same: completely random.

As Kauffman already noted [Kau93]: "*if a searcher jumps beyond the correlation length of the landscape, then whether or not this landscape is correlated, the searcher is encountering a fully uncorrelated random landscape*". Apparently this is the case here.

4.4 Conclusions

The global structure of an NK-landscape can be denoted by the correlation structure in the form of an AR(1) process:

$$y_t = c + \alpha_1 y_{t-1} + \varepsilon_t$$

where ε_t is white noise. This verifies the claim made by Weinberger that NK-landscapes are generic members of AR(1) landscapes [Wei90]. This AR(1) model is obtained by applying a time series analysis, the Box-Jenkins approach, to a time series of fitness values obtained by generating a random walk via neighboring points in the fitness landscape. Every landscape has its own specific values for the parameters of this model, which are estimated in the time series analysis. Using this model to describe the correlation structure of a fitness landscapes tells a lot about this structure:

- The AR(1) model implies that the fitness at a particular step (y_t) in a random walk generated on this landscape totally depends on the fitness one step ago (y_{t-1}), and some stochastic variable (ε_t). Knowing the fitness of two steps ago does not give any extra information about the expected value of the fitness at the current step.

- One of the properties of an AR(1) process is that the value of the parameter α_1 is the correlation coefficient between the fitness of two points one step apart in a random walk. The results show that this value decreases as K, the richness of epistatic interactions, increases. As a consequence, the correlation length of the landscape also decreases.

- The variance of the stochastic variable ε_t, also estimated in the time series analysis, indicates the amount of influence that this variable has in the model. As K increases, this variance increases, indicating a larger influence, and thus less correlation between the fitness of points one step apart.

- The value of R^2, a measure of goodness of fit of the model, indicates the explanatory and predictive value of the model. As K increases, this value decreases, indicating less explanatory and predictive value.

So, the correlation structure of different landscapes can be compared in terms of this AR(1) model. Furthermore, random walks can be generated with all sorts of genetic operators, which do not necessarily visit neighboring points in the landscape. The models obtained from an analysis of the data generated this way, indicate how other operators experience the correlation structure of a particular landscape. This appears indeed to be different from the actual structure.

Now that a way to determine and express the global structure of a fitness landscape is known, the flow of a population upon such a landscape can be examined, to see if this can be somehow related to the structure of this landscape. The next chapter presents the results of applying different search strategies to NK-landscapes, which can give more insight in such a relation.

Chapter 5

Population Flow

Adaptive evolution is a search process—driven by mutation, recombination, and selection—on a fitness landscape. An adapting population flows over the landscape under these forces. So, to gain more insight into the population flow on fitness landscapes, it is useful to apply different search strategies—based on mutation, recombination, and selection—to such landscapes.

This chapter presents the results of applying different search strategies, as introduced in Chapter 3, to different NK-landscapes. The performance of these strategies is evaluated by the performance measures also introduced in Chapter 3. First, the experimental setup is described. Next, the strategies are evaluated by each of the performance measures. Finally, some conclusions are drawn, and the validity of Kauffman's statement about the three time scales in adaptation (see Section 1.1) is assessed.

5.1 Experimental setup

The following search strategies (see Chapter 3 for the exact implementations) are applied to different NK-landscapes:

- Random ascent hillclimbing with memory (RAHCM)
- Random neighbor ascent hillclimbing (RNAHC)
- Long jumps
- Genetic Algorithm with one-point crossover (GA-ONEP)
- Hybrid Genetic Algorithm (HGA)

All strategies are allowed to do a total of 10,000 function evaluations. The different performance measures are recorded every 50 function evaluations.

NK-landscapes (see Chapter 2) with the following values for N and K are taken: N=100, K=0, 2, 5, 25, 50, and 99. In this chapter, only random interactions are considered. Furthermore, because an NK-landscape depends on randomly assigned fitness values, all results are averaged over 100 runs, each run on a different landscape but with the same values for N and K, to avoid statistical biases.

The next four sections each evaluate the different search strategies by one of the performance measures introduced in Chapter 3: maximum fitness, on-line performance, off-line performance, and mean Hamming distance.

5.2 Evaluation by maximum fitness

Figures 5.1 to 5.6 show the maximum fitness of the strategies plotted against the number of function evaluations for the different values of K. Note that for K=0 the Hybrid Genetic Algorithm is not applied, because the HGA would not be different from the RAHCM strategy (remember this strategy is also applied within the HGA): all individuals will climb to the one and only peak in the landscape, and crossover applied to a population with completely identical individuals makes no difference.

In the next subsections, the results with respect to the maximum fitness are evaluated per type of landscape.

5.2.1 Smooth landscapes: K=0

As Figure 5.1 shows, RAHCM finds the (only) optimum quickly, while RNAHC takes much longer to find this optimum. This is due to the many function evaluations the latter strategy has to do before it can step to a fitter neighbor: first, all neighbors are evaluated, and only then a fitter one is chosen. As the graph shows, this costs a lot of (wasted) evaluations.

The strategy of Long jumps is very poor. In fact it is nothing more than just a random search. Initially, it is able to find some fitter individuals quite quickly, but the more fitter individuals are found, the longer it takes to find yet another one. Kauffman already showed that there is a "universal law" for long jump adaptation: the waiting time to find a fitter individual doubles each time one is found [Kau93]. So, it is not to be expected that this strategy will perform well.

The maximum fitness in the GA-ONEP population increases steadily and quite fast, but it somehow seems to be unable to catch up with the RAHCM strategy. To find out whether this is just a statistical difference or a real discrepancy, a two-sample test for comparing the means of two stochastic variables is done. This test, using the mean and standard deviation of the samples, is described in Appendix A. The mean ($\overline{X}$) and the standard deviation (S) of the RAHCM, RNAHC and GA-ONEP strategies over the 100 runs at the 10,000th function evaluation are shown in Table 5.1.

	$\overline{X}$	S
RAHCM	0.670457	0.023997
RNAHC	0.668549	0.021999
GA-ONEP	0.663980	0.021260

Table 5.1: The mean $\overline{X}$ and standard deviation S of the maximum fitness over 100 runs at the 10,000th function evaluation.

The two-sample test is done to test for equality between the means of RAHCM and RNAHC and between the means of RAHCM and GA-ONEP. The values of the variables t_0 and v (see Appendix A) for both cases are shown in Table 5.2.

	t_0	v
RNAHC	0.58609	197
GA-ONEP	2.01779	195

Table 5.2: Test results of comparing RAHCM with two other strategies at the 10,000th function evaluation.

Taking a significance level α of 0.05 (so $1-\alpha/2$=0.975), and looking up the value of $t_{0.975}(v)$ for v = 195 and v = 197 in a table of the Student's t distribution[1], yields the following result: $1.960 < t_{0.975}(v) < 1.980$ for both values of v. Consequently, the hypothesis that the means for RAHCM and RNAHC are equal can <u>not</u> be rejected, because for this case $t_0 = 0.58609 < 1.960 < t_{0.975}(197)$. So, the small difference between RAHCM and RNAHC at the 10,000th function evaluation in Figure 5.1 is just a sampling error.

The results for the hypothesis that the means of RAHCM and GA-ONEP are equal, however, are less clear. For α=0.05 it holds that $t_0 = 2.01779 > 1.980 > t_{0.975}(195)$, but the difference is not really that large. Taking α=0.02, though, yields $t_0 = 2.01779 < 2.326 < t_{0.99}(195)$. So, for α=0.05 the hypothesis can be rejected (although not very convincing), but for α=0.02 it can not be rejected anymore.

Comparing the RAHCM and GA-ONEP strategies on one and the same landscape for K=0 shows that GA-ONEP also finds the (same) global optimum as RAHCM does, and that at the 10,000th function evaluation a large part of the population has converged to this optimum (results not shown). This makes clear that it can indeed <u>not</u> be concluded that the difference between RAHCM and GA-ONEP, shown in Figure 5.1, is a real difference; it is also a sampling error.

[1] Every book about statistics provides such a table

So, GA-ONEP does indeed find the global optimum, but it takes longer to find it than RAHCM does. On the other hand, it is much faster than RNAHC. These results, that a Genetic Algorithm is outperformed in speed by one specific type of hillclimbing, but is faster than other forms of hillclimbing on a smooth fitness landscape with only one optimum, is in accordance with results obtained in [MHF].

5.2.2 Rugged landscapes: K=2, 5

As Figures 5.2 and 5.3 show, it takes a little longer for RAHCM to find a good optimum on rugged landscapes than to find the global optimum on a smooth landscape, but even after 10,000 function evaluations the maximum for RAHCM is still slowly increasing, so once in a while an even higher hilltop is found. The cost that RNAHC has to pay to choose a fitter neighbor is too much on rugged landscapes: after 10,000 function evaluations RNAHC is still far behind on RAHCM. The strategy of Long jumps is again not better than just a random search. The universal law for long jump adaptation is in force here too.

The difference between GA-ONEP and RAHCM, compared with a smooth landscape, seems to be quite significant on rugged landscapes. Testing the hypothesis that the means over the 100 runs of these two search strategies are equal at the $10{,}000^{th}$ function evaluation, gives the results shown in Table 5.3.

	t_0	v
K=2	3.58490	180
K=5	12.42182	165

Table 5.3: Test results of comparing RAHCM with GA-ONEP at the $10{,}000^{th}$ function evaluation for K=2 and 5.

Even at a significance level α of 0.02, the hypothesis is rejected for both values of K. For K=2 it holds that $t_0 = 3.58490 > 2.358 > t_{0.99}(180)$, and for K=5 that $t_0 = 12.42182 > 2.358 > t_{0.99}(165)$. So, on rugged landscapes the difference is really significant.

It appears that GA-ONEP becomes trapped in a local region of the space where the fitnesses of the local optima are significantly less than those of the highest optima in the landscape. The crossover operator becomes less able to find the highest optima, because the correlation length for crossover is less on rugged landscapes than on smooth landscapes (see Section 4.3.2). Apparently, crossover is able to find a relatively good region in the landscape initially, but it is unable to structurally find even better regions.

The HGA, however, performs very well. It finds good optima relatively fast, and is also able to hold the population there. This, of course, is due to the stringent selection, which only holds the best individuals in the population (see Section 3.1.4). It appears that in the first part of the search the HGA does at least as well as the RAHCM strategy, but eventually the HGA stays constant, while the RAHCM strategy keeps increasing, although very slowly. Again an indication that crossover is not able to find better regions in the landscape in the long run, when the population has already converged to an initially found good region.

There is one striking feature about rugged landscapes: although the good optima are harder to find, they tend to be higher than the optimum in a smooth landscape. An explanation for this is given in appendix B.

5.2.3 Very rugged landscapes: K=25, 50

Figures 5.4 and 5.5 show that the two hillclimbing strategies show the same trend on very rugged landscapes as on rugged landscapes: it takes even longer to find good optima, and, during the rest of the search, better optima are found slowly. Also, RNAHC still has a disadvantage compared to RAHCM. Furthermore, Long jumps are again not very useful for finding good optima.

The GA-ONEP strategy still seems to be unable to reach better regions in the landscape in the long run, and gets stuck in intermediate regions. Even the HGA, although keeping up with RAHCM initially, gets stuck after a certain amount of evaluations. Crossover seems to be unable here to find better hillsides in the long run, not surprisingly considering the very low, or even absence of, correlation length for crossover on very rugged NK-landscapes with random interactions (see Section 4.3.2).

5.2.4 Completely random landscapes: K=99

On a completely random landscape, as shown in Figure 5.6, RAHCM, RNAHC and Long jumps perform equally well. All three search strategies boil down to just a random search. It does not matter anymore whether small or large jumps are made, the time to find fitter individuals is equally long.

Crossover appears to have no use at all on completely random landscapes, considering the performance of GA-ONEP and HGA. Information about the local structure at one place in the landscape implies nothing about the local structure in another place. On a completely random landscape, only random search is possible.

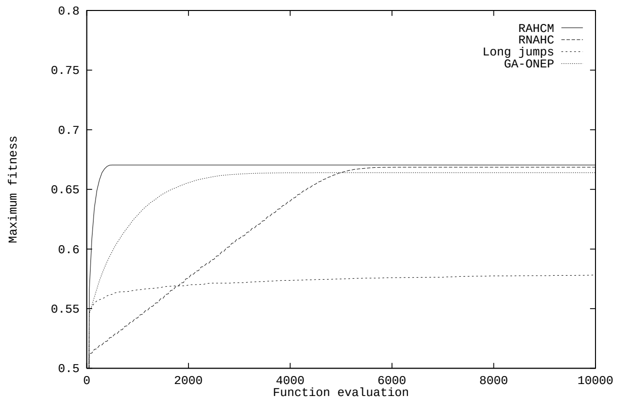

Figure 5.1: The maximum fitness of the search strategies for K=0.

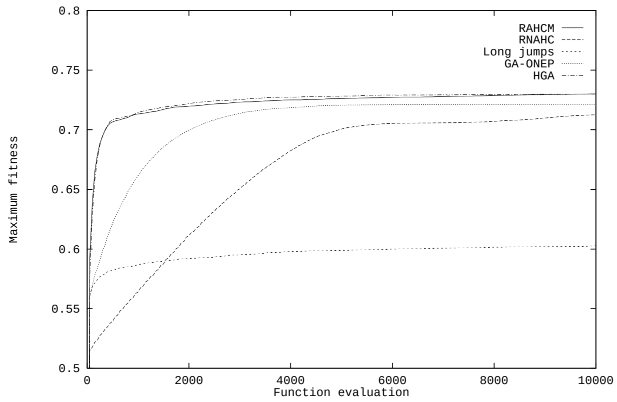

Figure 5.2: The maximum fitness of the search strategies for K=2.

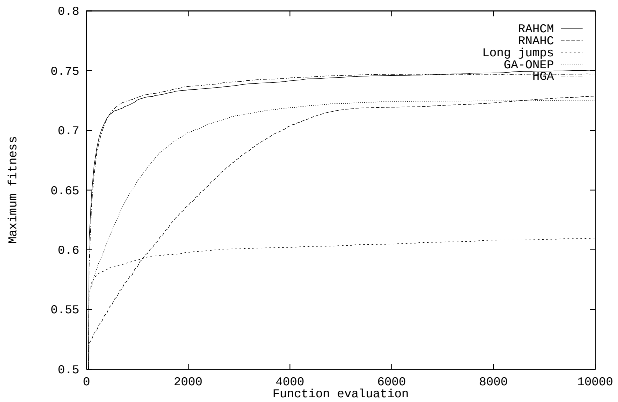

Figure 5.3: The maximum fitness of the search strategies for K=5.

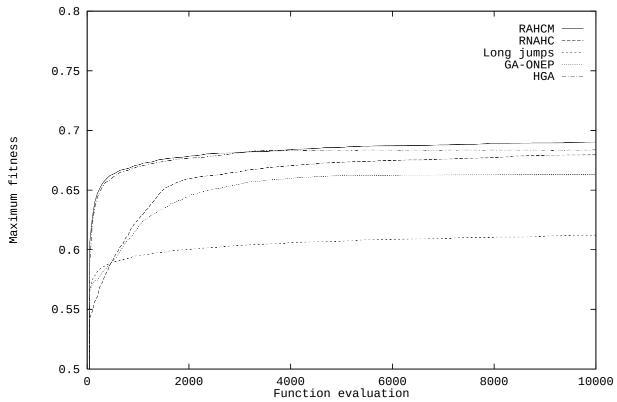

Figure 5.4: The maximum fitness of the search strategies for K=25.

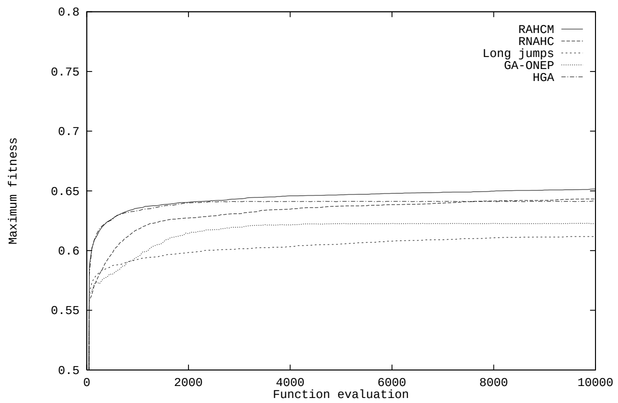

Figure 5.5: The maximum fitness of the search strategies for K=50.

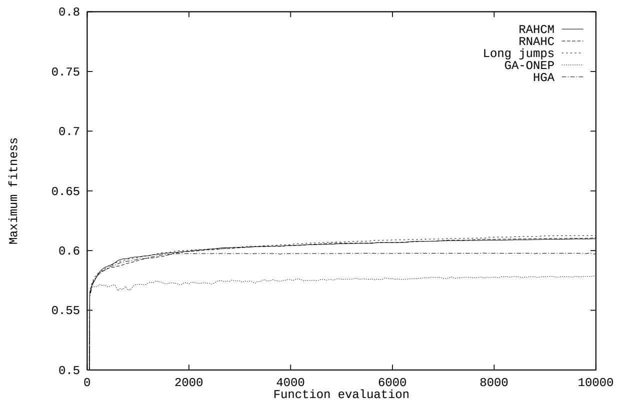

Figure 5.6: The maximum fitness of the search strategies for K=99.

5.3 Evaluation by on-line performance

The on-line performance of a search strategy gives an overview of the average value of all function evaluations done by this strategy up to a certain time. A high on-line performance means that the strategy is evaluating mostly good individuals, while a very low on-line performance means that it is wasting too much time on bad individuals.

On all landscapes, except the completely random one (K=99), the same sort of picture can be seen, as Figures 5.7 to 5.11 show. RAHCM initially increases quickly, but stays constant on an intermediate level for the rest of the search. This is due to the fact that each time an optimum is found, the search starts anew at a random point, and then gradually climbs up again. The high costs that RNAHC has to pay are shown here very clearly: the on-line performance increases very slowly and reaches only a moderate level. This shows again that the RNAHC strategy pays too much compared to the gain in fitness it receives.

That the strategy of Long jumps is really a random search, is shown by the fact that on all landscapes its on-line performance is exactly 0.5 throughout the search. So, on average, this strategy is evaluating just as much individuals that have a below-average fitness, as individuals that have an above-average fitness. There is no direction at all in the search.

A striking result is that, although GA-ONEP is not able to reach the highest peaks in (very) rugged landscapes, in the long run it outperforms all other strategies in on-line performance. So, it looks like GA-ONEP is a very efficient strategy, but this may be a little misleading. More will be said about this when the strategies are evaluated by population diversity.

The HGA performs initially just as well as RAHCM, but after a while it suddenly starts to become more efficient. This is the result of the stringent selection mechanism, which does not allow individuals that are less fit than the current members of the population to enter this population. So, this HGA never "falls back" to less good regions of the landscape, as RAHCM does each time it starts the search anew at a random point.

Completely random landscapes, shown in Figure 5.12, give a different picture. Apparently, all strategies, except GA-ONEP, are "degraded" to just random search. Only GA-ONEP manages to escape this fate, mainly due to its selection mechanism. The HGA strategy also uses a (very strong) selection mechanism, but here every generation all the neighbors of each genotype in the population are evaluated by the hillclimbing part of the algorithm, which, as said in Section 5.2.4, is just a random search when the landscape is completely random.

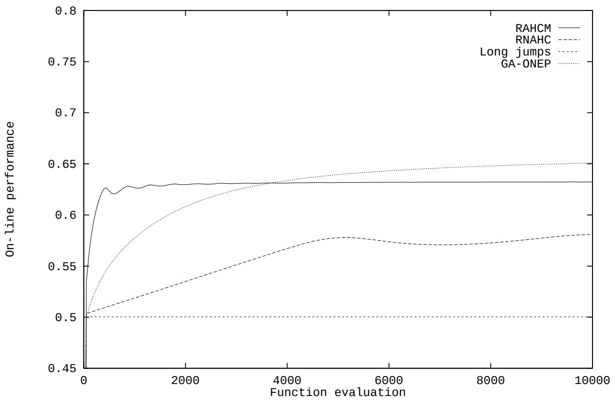

Figure 5.7: The on-line performance of the search strategies for $K=0$.

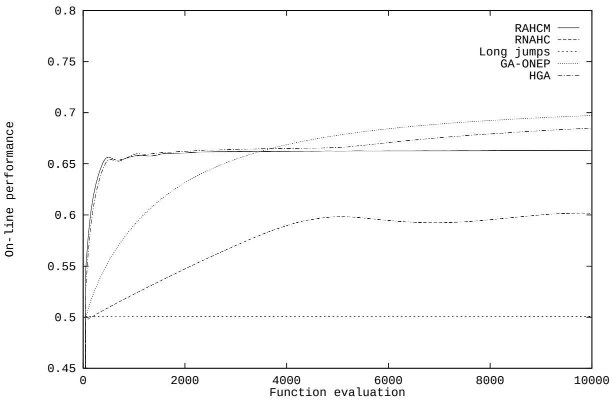

Figure 5.8: The on-line performance of the search strategies for $K=2$.

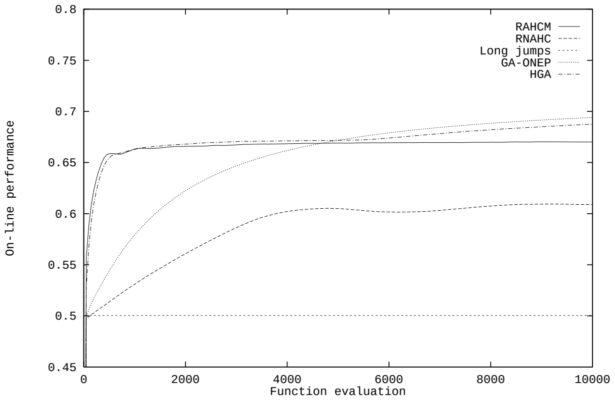

Figure 5.9: The on-line performance of the search strategies for $K=5$.

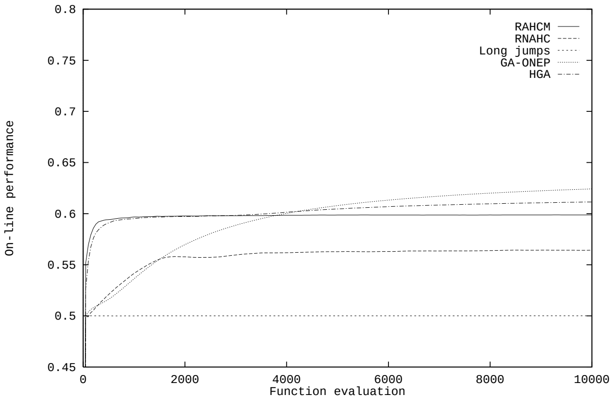

Figure 5.10: The on-line performance of the search strategies for $K=25$.

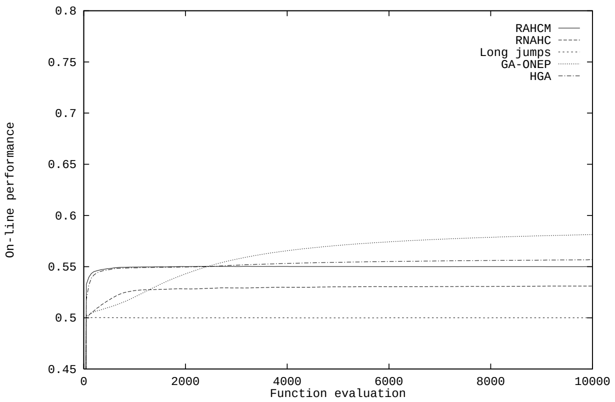

Figure 5.11: The on-line performance of the search strategies for $K=50$.

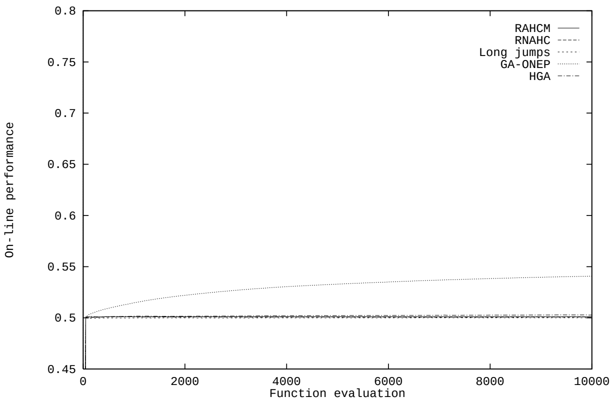

Figure 5.12: The on-line performance of the search strategies for $K=99$.

5.4 Evaluation by off-line performance

Figures 5.13 to 5.18 show the off-line performance of the strategies on all six landscapes. In the off-line performance, the best fitness found up to a certain time is averaged, so this is independent of the fitness of the individuals in the population at that time.

The general picture of the off-line performance shows not much difference with that of the maximum fitness, so it does not contribute any new information. It once more shows that RNAHC converges much too slow to an optimum, and that the Genetic Algorithm is outperformed in speed by the strong hillclimbing scheme RAHCM on smooth fitness landscapes.

5.5 Evaluation by mean Hamming distance

To gain more insight into the diversity of a population during the search, the mean Hamming distance for the population-based strategies is shown in Figures 5.19 to 5.24.

The members in the population of long jumpers do not converge at all, except a little for the smooth (K=0) landscape. In such a landscape there is only one hill, so every jump to a fitter individual brings the members of the population a little closer to the one and only optimum, and thus a little closer to each other.

The GA-ONEP population, however, converges rather quickly (except on a completely random landscape). After some time, the members of the population differ, on average, in just one or two bits. The population then becomes a tight cluster in the fitness landscape. This partly explains the seemingly efficient behavior of GA-ONEP that can be observed in the on-line performance (see Section 5.3): GA-ONEP is doing a lot of the same, "relatively good", function evaluations, because the members of the population are stuck in a small region of intermediate optima in the fitness landscape. This strong convergence could indicate that either the selection pressure is too high or that the mutation rate is too low, or maybe both.

By contrast, the HGA population is hardly converging. This shows that, although crossover is applied, all individuals are climbing different hillsides. Only for K=2, the population converges to some extent. This is due to a special structure that appears to exist in these landscapes, called a massif central. This feature is discussed in the next chapter.

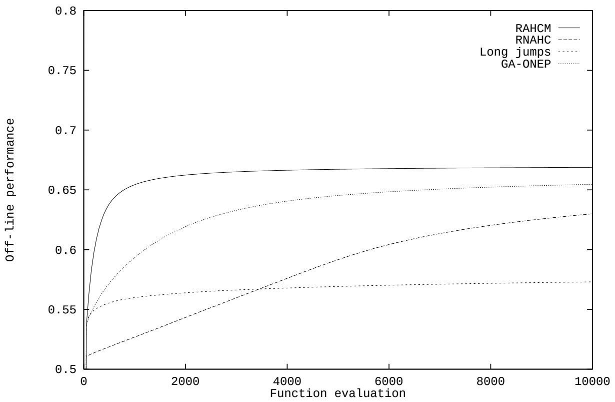

Figure 5.13: The off-line performance of the search strategies for K=0.

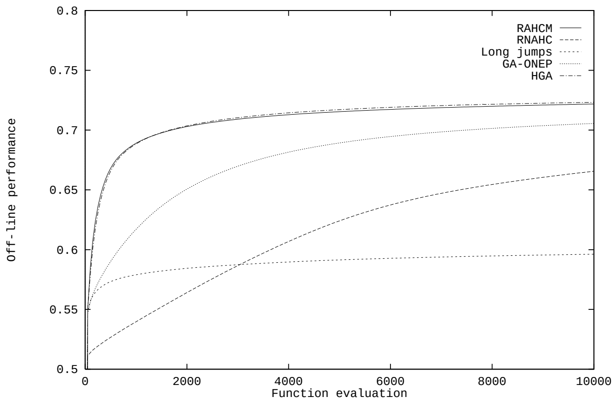

Figure 5.14: The off-line performance of the search strategies for K=2.

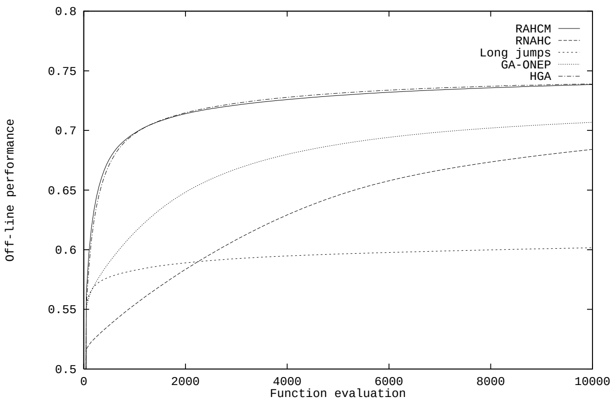

Figure 5.15: The off-line performance of the search strategies for K=5.

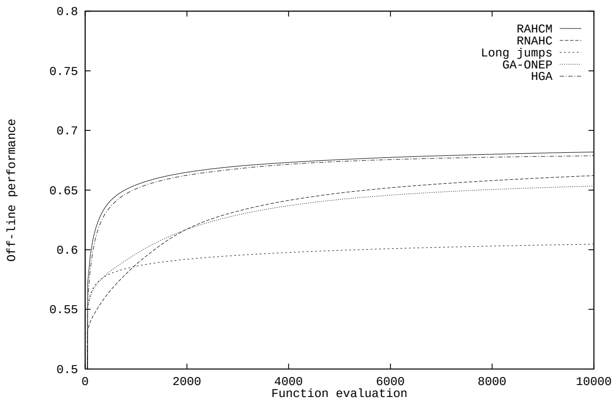

Figure 5.16: The off-line performance of the search strategies for K=25.

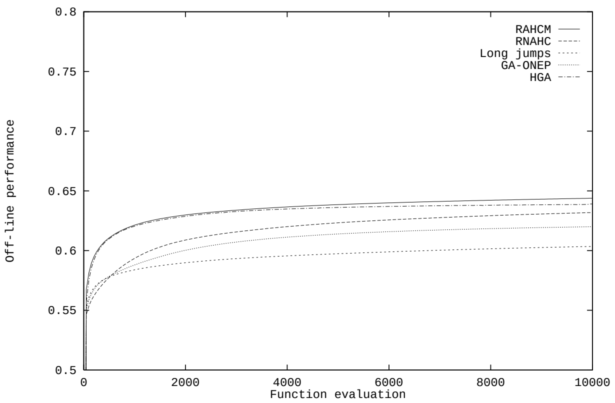

Figure 5.17: The off-line performance of the search strategies for K=50.

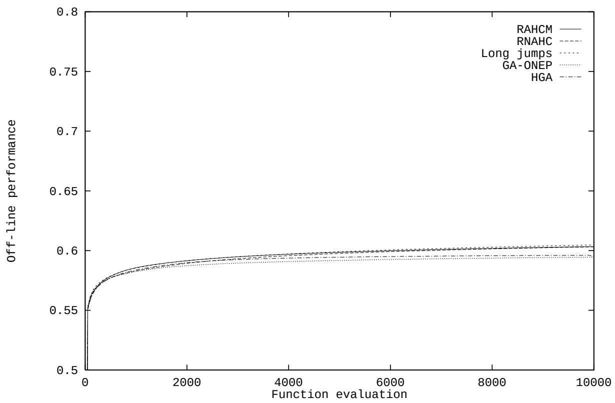

Figure 5.18: The off-line performance of the search strategies for K=99.

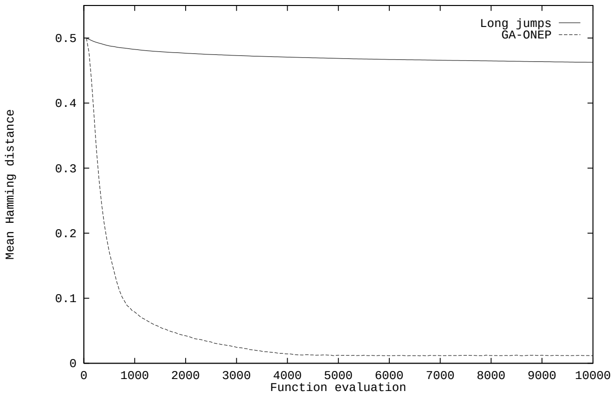

Figure 5.19: The mean Hamming distance of the search strategies for K=0.

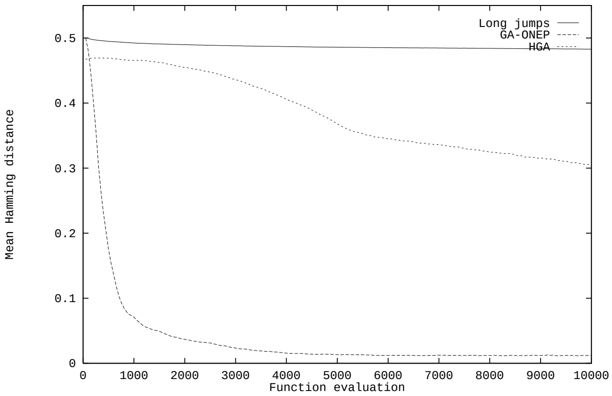

Figure 5.20: The mean Hamming distance of the search strategies for K=2.

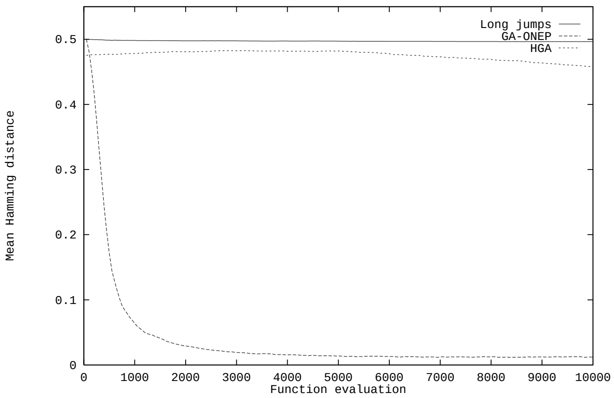

Figure 5.21: The mean Hamming distance of the search strategies for K=5.

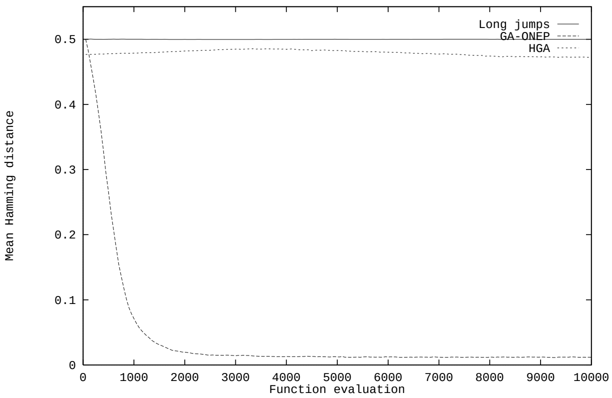

Figure 5.22: The mean Hamming distance of the search strategies for K=25.

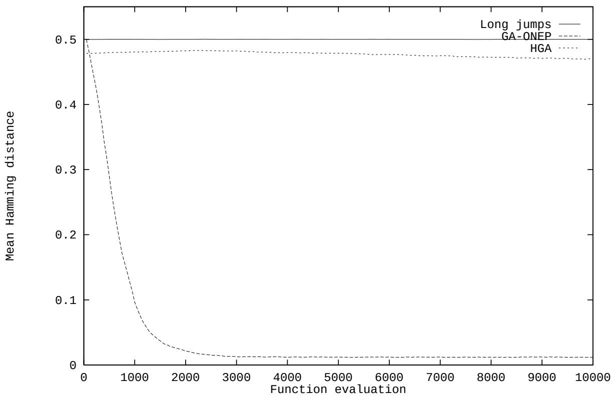

Figure 5.23: The mean Hamming distance of the search strategies for K=50.

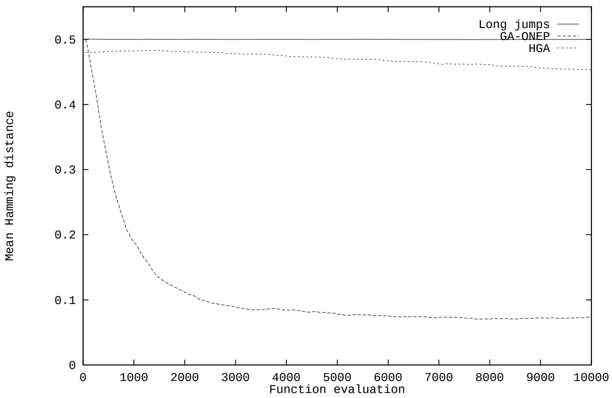

Figure 5.24: The mean Hamming distance of the search strategies for K=99.

5.6 Conclusions

First, the general conclusions about population flow that can be drawn from the results are given. Next, the validity of Kauffman's statement about the three time scales in adaptation on rugged fitness landscapes is assessed. After that, some implications that follow from the conclusions are discussed. Finally, a summary of the main conclusions is given.

5.6.1 General conclusions

The five search strategies that are applied to NK-landscapes all show a different type of behavior. Next, a summary of their performance is given:

- As already observed in Section 3.1.1, it indeed matters which type of hillclimbing is used. RAHCM appears to work very well on all types of landscapes, from smooth to completely random. On the other hand, RNAHC is much too slow in finding fitter individuals. Hillclimbing is not a really "efficient" strategy, because it spends a substantial amount of evaluations on less fit individuals, but it is able to find good optima quite fast.

- Long jumps is nothing more than just a random search, which only works well on completely random landscapes. As Kauffman already showed, there exists a universal law for Long jumps: the time to find a fitter individual doubles each time one is found.

- The Genetic Algorithm is able to find a relatively good region in the fitness landscape initially, but in the long run the population converges and gets stuck in such a region, preventing still better regions to be found. It seems to be a very efficient algorithm, but this is merely due to this strong convergence property. The more rugged a landscape becomes, the less useful crossover will be due to the lack of correlation for this operator.

- The Hybrid Genetic Algorithm performs very well on most landscapes, but eventually gets stuck on local optima just below the highest peaks in the landscape. For this algorithm too, crossover becomes less useful on more rugged landscapes.

Local search (like hillclimbing) is useful on all types of landscapes, but the speed at which fitter individuals are found depends on the exact implementation of the local search strategy. A local search strategy makes use of information from neighboring points in the landscape to direct the search to a nearby hilltop. Since for almost all types of landscapes (except for completely random landscapes) there exists correlation between the fitness of neighboring points (see Section 4.3.1), this strategy works very well. On completely random landscapes, the local search itself just comes down to a random search, which is the only possibility on these types of landscapes.

Global search (like crossover), on the other hand, appears to be most useful on smooth and not-too-rugged landscapes (K=0, 2, 5). On these landscapes, operators like one-point crossover experience enough correlation between the fitness of parents and offspring (see Section 4.3.2) to direct the search to a relatively good region in the landscape. Once such a good region has been found, however, local search (like mutation, or hillclimbing) has to "fine tune" the population to the highest peaks within this region; global search like crossover is not able to do this by itself.

So, global search should always be combined with local search, but also with some form of selection, to prevent this global search to become just a random search (only the best individuals should be used to direct the global search). Using a form of selection, however, restricts the evolvability of a population: once a relatively good region in the landscape is found, selection will tend to keep the population on the peaks within this region, preventing other, maybe better, regions to be found.

In [Kau93] it is stated that: "*Whether both evolvability and sustained fitness can be jointly optimized is unclear*". The results obtained in this chapter for the Genetic Algorithm can not give a positive answer to this question. The (strong) selection ensures sustained fitness, but limits evolvability. On the other hand, when the individual runs are considered, it appears that sometimes a better individual is found which cannot be kept in the population (results not shown). When the selection becomes weaker, or the mutation rate becomes higher, this will happen more often, thus threatening sustained fitness in favor of evolvability. In this case, the error catastrophe will occur (selection is not able to hold the adapting population on the highest peaks; see page 13). So, it will be difficult, if not impossible, to find a good selection pressure together with a good mutation rate that avoids both the error catastrophe and premature convergence on rugged fitness landscapes.

Random search (like Long jumps) appears to work well on completely random landscapes only. Not surprisingly considering the fact that long jump adaptation experiences no correlation at all on a fitness landscape, no matter what the actual correlation structure of this landscape is (see Section 4.3.3). Random search can probably be useful in combination with global search, for example when a population becomes stuck on the peaks of one particular region in a fitness landscape. Long jumps can then be helpful in finding other regions in the landscape that might contain even higher peaks.

5.6.2 Time scales in adaptation

Kauffman identifies three natural time scales in adaptation on rugged fitness landscapes [Kau93]:

1. Initially, fitter individuals are found faster by long jumps than by local search. However, the waiting time to find such fitter individuals doubles each time one is found.

2. Therefore, in the midterm, adaptation finds nearby fitter individuals faster than distant fitter individuals and hence climbs a local hill in the landscape. But the rate of finding fitter nearby individuals first dwindles and then stops as a local optimum is reached.

3. On the longer time scale, the process, before it can proceed, must await a successful long jump to a better hillside some distance away.

So, Kauffman states that initially long jumps find fitter individuals faster than local search does. He used RNAHC as local search strategy, which is indeed slower than long jumps (see Figures 5.1 to 5.6). However, it was already stated in Section 3.1.1, that in comparing other search strategies with hillclimbing, it matters *which* type of hillclimbing algorithm is used. This statement is clearly validated by the results obtained in this chapter. It appears that, taking RAHCM as local search strategy, the first time scale does not hold anymore. (see again Figures 5.1 to 5.6).

Furthermore, there seems to be no directional search in Kauffman's first phase. Distant points are tried at random in the hope that one of them will have a higher fitness than the current highest fitness in the population. But as the results in this chapter show, directed global search can be very useful in finding good regions in the landscape initially.

The general picture about the flow of a population on a fitness landscape that the results obtained in this chapter provide, looks more like a process of only *two* time scales, or phases, in adaptation:

1. Initially, a relatively good region in the landscape is found by a (directed) global search.

2. On the longer time scale, local search has to "fine tune" the adapting population by finding the best peaks within this relatively good region.

So, the results presented here do not support Kauffman's statement about three time scales in adaptation, but imply an adjusted one that incorporates only two time scales, or phases.

5.6.3 Some implications

Different implications follow from the above conclusions, depending on what point of view is taken. From the viewpoint of problem solving, the main interest is in finding a good solution for a problem as fast as possible. A strong hillclimbing algorithm appears to perform this task quite well on all types of landscapes. A global search strategy (for example crossover) combined with a local strategy (for example hillclimbing) works well too, but mainly on not-too-rugged landscapes, on which there is still enough correlation experienced by the global search operator.

In using a Genetic Algorithm for problem solving, a good balance has to be found between evolvability and sustained fitness, that is, between mutation rate and selection pressure. A mutation rate that is too low compared to the selection pressure causes premature convergence, while a mutation rate that is too high compared to the selection pressure gives rise to the error-catastrophe. Furthermore, a GA on its own is probably not powerful enough to find the best solutions, but several studies indicate that using problem-specific information in the algorithm can enhance its performance substantially (see for example [ERR94, MHF]).

From a biological viewpoint, more care has to be taken in finding valid implications. Not all search strategies are biologically plausible. In the RAHCM strategy for example, no neighbor is evaluated more than once, while in the RNAHC strategy, all neighbors are evaluated at each step. In Nature, something in between happens: some, but not necessarily all, neighbors are evaluated, some of which even more than once. The Genetic Algorithm is a rather plausible model of a real adapting population, and even the strategy of the Hybrid Genetic Algorithm occurs in Nature (see [MS93]). The Elitist Recombination operator, however, is rather artificial (usually, parents do not throw away their children if they appear to be less fit, whatever that means, than they are themselves).

Another simplification in the experiments done here, compared to Nature, is that the fitness landscape stays fixed during the search, or evolution. In Nature, the environment (which determines the fitness landscape) is constantly changing, as a result of this evolution. But opposed to these drawbacks, the advantage of using evolutionary models is that some basic aspects of evolution can be isolated and studied in detail, which is much more difficult in Nature itself. So, it will still be possible to derive some biologically plausible implications from the results presented in this chapter.

One of these implications follows from the result that crossover is most useful on not-too-rugged landscapes. The more complex organisms in Nature, including humans, reproduce sexually, which involves the crossing over of the genetic material of both parents. It should therefore be expected that the fitness landscapes on which these organisms evolve are not-too-rugged, allowing this crossover to be useful in the evolution. This means that the amount of epistatic interactions is relatively small, compared with the length of the genotypes (in the order of $0.05N$ for example, N being the length of the genotype). On such landscapes, crossover still experiences enough correlation to be able to find good regions in these landscapes.

Another implication is, that a population or a species should keep some evolvability within the population or species. Otherwise, there is a danger of becoming trapped inside a small region of the landscape, having lost the ability to find other, maybe better, regions. If a species is no longer able to escape from such a small region in the landscape, this species might become extinct. Nature seems to have found some solutions against this danger, for example by trying to avoid inbreeding.

A final implication is, that a lot can still be learned from Nature. The models and search strategies, as used in this thesis, do not by a long way reach the complexity that can be seen in Nature itself. There still has to be done a lot of work for a complete understanding of the processes that drive the flow of a population on a fitness landscape.

5.6.4 Summary

Local search (like Hillclimbing) is useful on all types of landscapes, while global search (like crossover) is most useful on smooth and not-too-rugged landscapes. Global search is able to find a relatively good region in the landscape initially, but it is unable to "fine tune" a population to the highest peaks within this region. Therefore, global search should always be combined with local search.

The simultaneous optimization of evolvability and sustained fitness remains a problem. Selection should be strong enough to ensure sustained fitness, but not too strong, because this limits evolvability. The results of this chapter can not give a definitive answer to whether, or how, this problem can be solved.

The three times scales Kauffman identifies in an adaptive search, are not supported by the results of this chapter. Instead, a picture of only two time scales, or phases, in an adaptive search emerges. First, a relatively good region in the fitness landscape is found by global search. In the second phase, local search finds the highest peaks within this region.

In this chapter, only NK-landscapes with random interactions are considered, because this is the most general case. In fact, for most operators, like mutation and long jumps, it does not make any difference whether the epistatic interactions are randomly distributed or nearby. Only for operators that make use of recombination, like crossover, there is a difference. Therefore, the next chapter examines more thoroughly the usefulness of recombination in these different circumstances.

Chapter 6

The Usefulness of Recombination

In the previos chapter it was concluded that crossover, a form of recombination, is most useful on not-too-rugged landscapes, that is, landscapes with low epistasis. When the landscape becomes more rugged, the usefulness of crossover decreases. One explanation for this, is the small, or even absence of, correlation for crossover on very rugged landscapes.

In this chapter, the usefulness of recombination is examined more thoroughly. First, the relation between the type of recombination that is used and the type of epistatic interactions on the fitness landscape is examined. Next, the usefulness of recombination in relation to the location of optima in the fitness landscape is examined. Finally, the conclusions that are drawn from the examination of these two relations are summarized, and the validity of Kauffman's statement about the usefulness of recombination (see Section 1.1) is assessed.

6.1 Crossover disruption

According to the building block hypothesis, a Genetic Algorithm works well when short, low-order, highly fit schemata (building blocks) are recombined to form even more highly fit higher-order schemata (see Section 3.1.3). So, a GA works well when crossover is able to recombine building blocks to longer, higher-order schemata with a high fitness. On the other hand, it follows from the Schema Theorem that long, high-order schemata are more affected by crossover disruption than short, low-order ones (see also Section 3.1.3). So, opposed to the usefulness of crossover in constructing longer, highly fit schemata, there is the danger of disrupting them again.

To investigate this construction-disruption duality, two types of crossover, one-point and uniform (see Section 3.1.3), are compared on fitness landscapes with different types of epistatic interactions, random and nearest neighbor. Uniform crossover is believed to be maximally disruptive, while one-point crossover is more conservative. But this depends highly on the type of epistatic interactions within a genotype, or bit string.

One-point crossover is more disruptive when the epistatic interactions in a genotype are randomly distributed than when they are the nearest neighbors: with random interactions almost every possible crossover point will affect the epistatic relations of almost all bits in a bit string, while with nearest neighbor interactions only the epistatic relations of the bits in the vicinity of the crossover point are affected. This is shown in Section 4.3.2 by the fact that one-point crossover has a larger correlation coefficient on landscapes with nearest neighbor interactions than on landscapes with random interactions.

With uniform crossover, however, there is a large chance that a good configuration of neighboring epistatically interacting bits will be disrupted. On the other hand, when the epistatic interactions are random, uniform crossover can recombine good values for these interacting bits, while one-point crossover is unable to do this.

Next, the experimental setup for examining the relation between the type of recombination that is used and the type of epistatic interactions on the fitness landscape is described, after which the results of the experiments are presented.

6.1.1 Experimental setup

Two Genetic Algorithms, having different crossover operators, are applied to different NK-landscapes (see Section 3.1.3 for the exact implementations of the GA's) :

- A GA with one-point crossover (GA-ONEP)
- A GA with uniform crossover (GA-UNIF)

The GA's have exactly the same implementation, except for their crossover methods. Both GA's are allowed to do a total of 10,000 function evaluations. Every 50 function evaluations (that is, every generation), the maximum fitness in the population is recorded.

NK-landscapes with the following values for N and K are taken: N=100, K=0, 2, 5, 25, 50, and 99. Both random and nearest neighbor interactions are considered. All results are averaged over 100 runs, each run on a different landscape, but with the same values for N and K.

6.1.2 Results

Figures 6.1 to 6.6 show the results of applying the two GA's to the different NK-landscapes. The abbreviation RND stands for random interactions, while NNI stands for nearest neighbor interactions. So, GA-ONEP (RND) means the Genetic Algorithm with one-point crossover applied to an NK-landscape with random epistatic interactions. Note that for K=0 and K=99 only random interactions are considered, because nearest neighbor interactions is exactly the same as random interactions for these values of K (see Section 2.3.1).

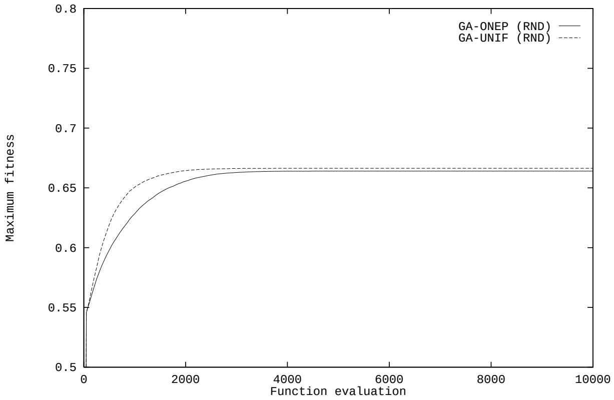

Figure 6.1: The maximum fitness of the two GA's for K=0.

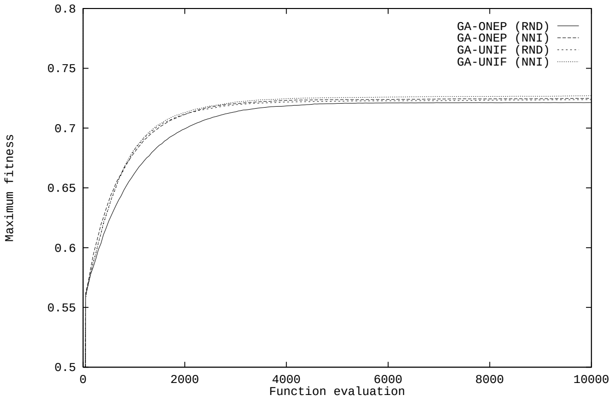

Figure 6.2: The maximum fitness of the two GA's for K=2.

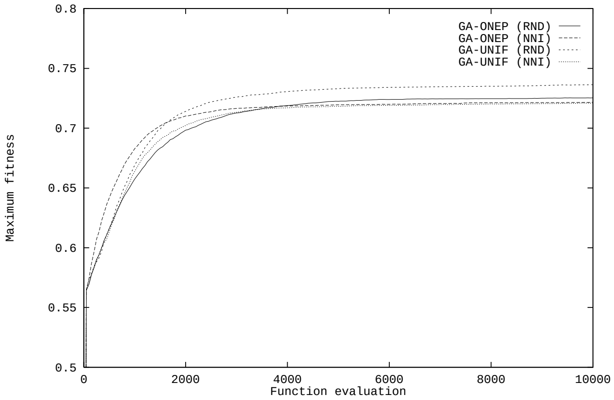

Figure 6.3: The maximum fitness of the two GA's for K=5.

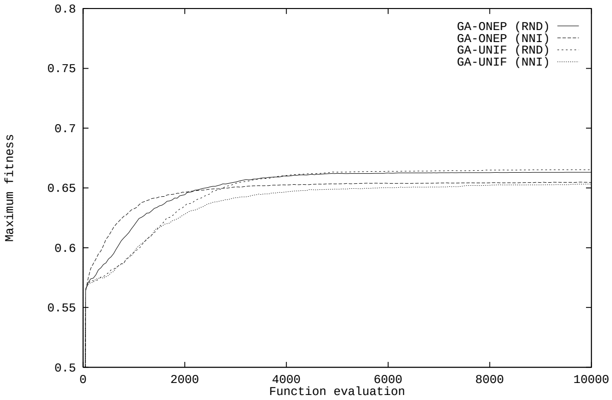

Figure 6.4: The maximum fitness of the two GA's for K=25.

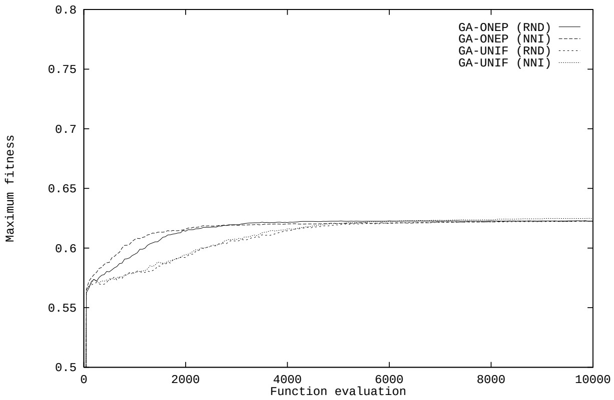

Figure 6.5: The maximum fitness of the two GA's for K=50.

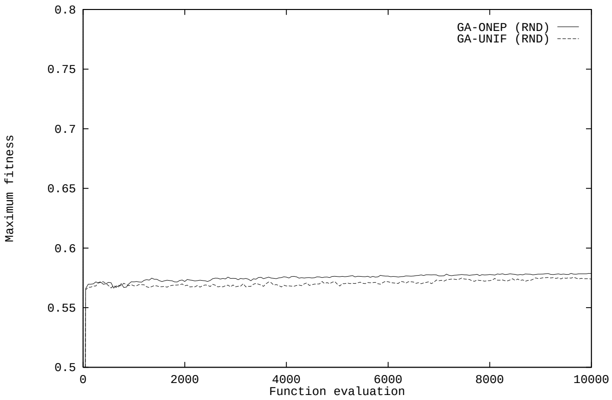

Figure 6.6: The maximum fitness of the two GA's for K=99.

The graphs clearly show the two phases in an adaptive search that are identified in Chapter 5: the first phase consists of finding a good region in the fitness landscape by global search, and the second phase consists of trying to find the highest peaks within this region by local search. Initially, the graphs are increasing rapidly (the first phase), but then they become gradually less steep (the second phase) until they are completely smooth, indicating that the highest peaks in a relatively good region are found. Only the completely random landscape (K=99, Figure 6.6) does not fit into this picture, because global search is useless on this landscape. Therefore, the case of K=99 is left out of the rest of the analysis.

Since the second phase in the search is dominated by local search, only the performance in the first phase of the search is evaluated here to examine the usefulness of recombination. Differences in performance in the second phase are mainly a reflection of differences in performance in this first phase. Furthermore, the results are viewed in two ways: taking one type of GA and comparing random with nearest neighbor interactions, and taking one type of epistatic interactions and comparing GA-ONEP with GA-UNIF.

Table 6.1 presents the results from the first viewpoint. It shows for both types of GA's on which type of landscape (that is, with random interactions (RND) or with nearest neighbor interactions (NNI)) they are better able to find a good region in the landscape in the first phase of the search. "Better able" means either finding such a region faster, or finding a better region (that is, containing higher peaks), or sometimes both. An X means that there is no difference between the two interactions.

	K=0	K=2	K=5	K=25	K=50
GA-ONEP	-	NNI	NNI	NNI/RND	NNI
GA-UNIF	-	X	RND	RND	X

Table 6.1: Comparison of random interactions (RND) with nearest neighbor interactions (NNI) for GA-ONEP and GA-UNIF in the first phase of the search. An entry RND means that the GA works better on a landscape with random interactions than on a landscape with nearest neighbor interactions. An X means that there is no difference.

The table shows that one-point crossover (GA-ONEP) works better on a landscape with nearest neighbor interactions than on a landscape with random interactions. So, one-point crossover is better able to combine configurations of nearby interacting bits (without disrupting them too much again), than configurations of random interactions. The entry NNI/RND for K=25 reflects the fact that the graph of GA-ONEP is initially increasing faster for landscapes with nearest neighbor interactions, but is overtaken by random interactions, for which it eventually finds a better region (that is, containing higher peaks), as can be seen in Figure 6.4.

The table shows furthermore that uniform crossover (GA-UNIF) works better with random interactions on NK-landscapes with intermediate epistasis (K=5 and K=25). Apparently, for very low and very high epistasis, uniform crossover is just as disruptive, no matter whether the epistatic interactions are randomly distributed or nearby.

Table 6.2 presents the results from the second viewpoint. It shows for both types of epistatic interactions which type of GA (GA-ONEP or GA-UNIF) is better able to find a good region in the landscape in the first phase of the search ("better able" in the same sense as in Table 6.1). Again, an X means no difference.

	K=0	K=2	K=5	K=25	K=50
RND	GA-UNIF	GA-UNIF	GA-UNIF	GA-ONEP	GA-ONEP
NNI	-	X	GA-ONEP	GA-ONEP	GA-ONEP

Table 6.2: Comparison of GA-ONEP with GA-UNIF for random interactions (RND) and nearest neighbor interactions (NNI) in the first phase of the search. An entry GA-ONEP point means that one-point crossover works better on that particular landscape than uniform crossover. An X means that there is no difference.

The table shows that for smooth and rugged landscapes (K=0, 2 and 5) uniform crossover (GA-UNIF) works better than one-point crossover (GA-ONEP) when the epistatic interactions are random. So, for low, random epistasis, uniform crossover is better able to combine building blocks, without disrupting them too much again, than one-point crossover. On the contrary, one-point crossover (GA-ONEP) works better for very rugged landscapes (K=25 and 50) when the epistatic interactions are random. So, for high random epistasis, uniform crossover becomes too disruptive.

For nearest neighbor interactions it appears that one-point crossover (GA-ONEP) works better than uniform crossover (GA-UNIF), except for K=2, where there is no difference. As expected, uniform crossover is too disruptive, compared with one-point crossover, when the epistatic interactions are nearby.

So, these results show clearly that there is a relation between the type of recombination that is used (one-point crossover or uniform crossover) and the type, and also the amount, of epistatic interactions (random or nearest neighbor) on the landscape. The next section examines the relation between the usefulness of recombination and the location of optima in the fitness landscape.

6.2 Recombination and the location of optima

The first condition that has to be met, according to Kauffman, for recombination to be useful, is that the high peaks in the landscape are near one another and hence carry mutual information about their locations in the fitness landscape. To examine to what extent this condition holds, the two types of crossover, one-point and uniform, are compared to each other and to a situation in which no crossover is applied. This is done on a fixed fitness landscape, of which the locations of local optima, relative to each other, are determined. Both random interactions and nearest neighbor interactions are considered.

6.2.1 Experimental setup

First, a fixed NK-landscape is generated for N=100 and K=2, both for random interactions and for nearest neighbor interactions. Three forms of an iterated hillclimbing strategy are then applied to each of these two landscapes: one with one-point crossover (IHC-ONEP), one with uniform crossover (IHC-UNIF), and one without crossover (IHC). The implementations of these strategies are as follows: (For the implementation of random ascent hillclimbing with memory, see section 3.1.1).

IHC-ONEP

1. Create a population of bit strings at random
2. Let all the members of the population (one after another) climb to a nearby hilltop, using random ascent hillclimbing with memory.
3. Shuffle the population in a random order. Apply one-point crossover to every next pair of parents in the population.
4. Repeat steps 2 and 3 for a set number of function evaluations.

IHC-UNIF

1. Create a population of bit strings at random
2. Let all the members of the population (one after another) climb to a nearby hilltop, using random ascent hillclimbing with memory.
3. Shuffle the population in a random order. Apply uniform crossover to every next pair of parents in the population.
4. Repeat steps 2 and 3 for a set number of function evaluations.

IHC

1. Create a population of bit strings at random
2. Let all the members of the population (one after another) climb to a nearby hilltop, using random ascent hillclimbing with memory.
3. Repeat steps 1 and 2 for a set number of function evaluations.

A population size of 10 is taken, and the crossover rate is 1.0, so crossover is always applied. The three strategies are all allowed to do 50,000 function evaluations. During the run, the maximum fitness in the population is recorded every 50 function evaluations.

Furthermore, the locations of the local optima, relative to each other, are determined for both landscapes, by applying random ascent hillclimbing with memory 10,000 times to each landscape. Every found local optimum is recorded, together with its fitness. The fitness of the local optima is then plotted against the (normalized) Hamming distance from the best found local optimum.

6.2.2 Results

Figures 6.7 and 6.8 show the results of applying the three search strategies, IHC-ONEP, IHC-UNIF, and IHC, to the fixed fitness landscape for random and nearest neighbor interactions, respectively. It is clear that on the landscape with random interactions both crossover operators are useful. If crossover is applied to the population, the maximum fitness in the population stays relatively high, indicating that the locations of two optima give information about the locations of other optima. Also, the graphs appear to be gradually increasing. There is not much difference between one-point and uniform crossover.

For nearest neighbor interactions, however, the distinction is less clear. The IHC-ONEP and IHC-UNIF strategies appear to be just a little better than the IHC strategy during the search, but not much. Crossover contributes just a little in finding good regions in the landscape. Furthermore, the graphs are certainly not increasing, but instead appear to decrease a little after a while. Again, there is not much difference between one-point and uniform crossover.

Figures 6.9 and 6.10 show the locations of local optima, relative to the best optimum found, for the landscapes with random interactions and nearest neighbor interactions, respectively. For the landscape with random interactions 9,970 different local optima were found, while for the landscape with nearest neighbor interaction 10,000 different local optima were found.

There is a clear similarity between the two plots. The optima with a relatively higher fitness tend to be closer to the best optimum than optima with a relatively lower fitness.

This shows a feature of the landscapes that Kauffman called a *massif central*: there is one place in the landscape where all the good optima are situated, surrounded by the less good optima. This feature is the cause that crossover can help in finding a good region in the landscape: recombining the information of two optima gives a high chance of finding still better optima.

This massif central is also the explanation for the fact that the population of the HGA strategy converges to some extent on K=2 landscapes (see Section 5.5). All individuals in the population of the HGA climb different hillsides, but by applying crossover to them, they all move closer and closer to the center of this massif central, and thus to each other. In landscapes with higher values of K, the optima are more or less randomly distributed (see [Kau93]). Therefore, on these kinds of landscapes, this convergence does not happen.

Besides the similarity, there is also one striking difference between the plots, though. For random interactions, the good optima are *much closer* to the best optimum (and thus to each other) than for nearest neighbor interactions. The number of optima with a (normalized) Hamming distance of 0.10 or less from the best optimum is 48 for random interactions, while it is only 2 for nearest neighbor interactions. This explains the difference in crossover performance (relative to no crossover) between the two landscapes. For nearest neighbor interactions, the good optima are just a little too far from the best optimum (and probably also from each other), to give enough information about the location of the highest peaks.

Kauffman already did this landscape analysis himself, and the plots shown here are very similar to his plots, which also show the similarity between random and nearest neighbor interactions. But because he used different scales for both plots, the striking difference between the two is much harder to detect. At least, Kauffman does not say anything about it.

So, these results make it clear that the high peaks in the landscape indeed have to be near one another, to make recombination really useful.

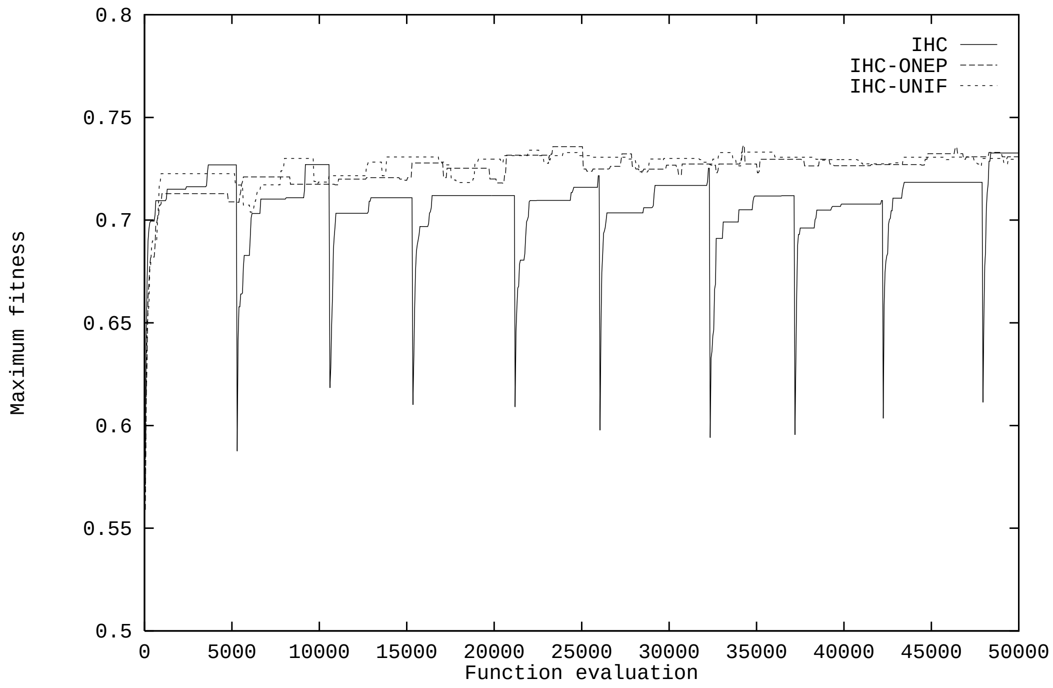

Figure 6.7: Comparison of 1-point (IHC-ONEP) and uniform (IHC-UNIF) crossover and no crossover (IHC) for K=2, random interactions.

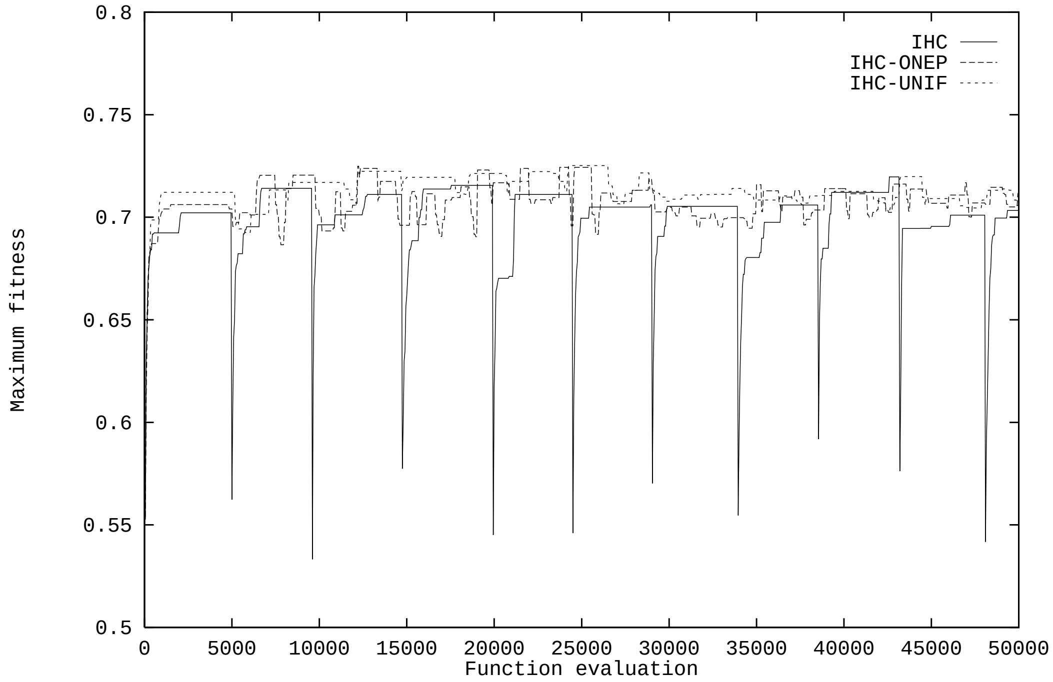

Figure 6.8: Comparison of 1-point (IHC-ONEP) and uniform (IHC-UNIF) crossover and no crossover (IHC) for K=2, nearest neighbor interactions.

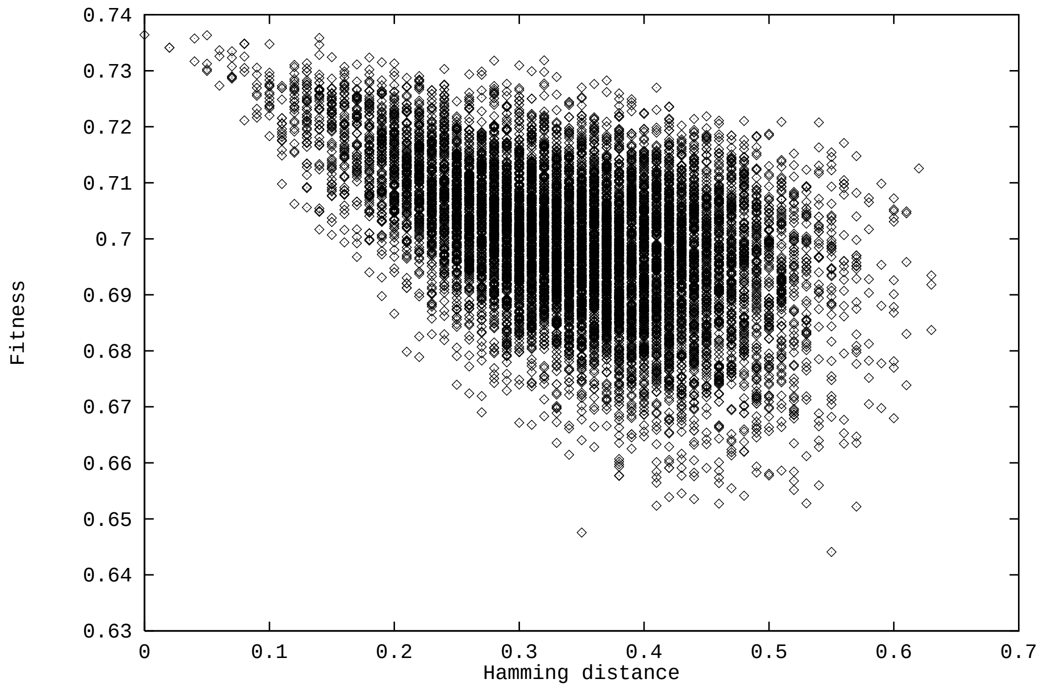

Figure 6.9: The correlation between the fitness of local optima and their (normalized) Hamming distance from the fittest local optimum found. K=2, random interactions.

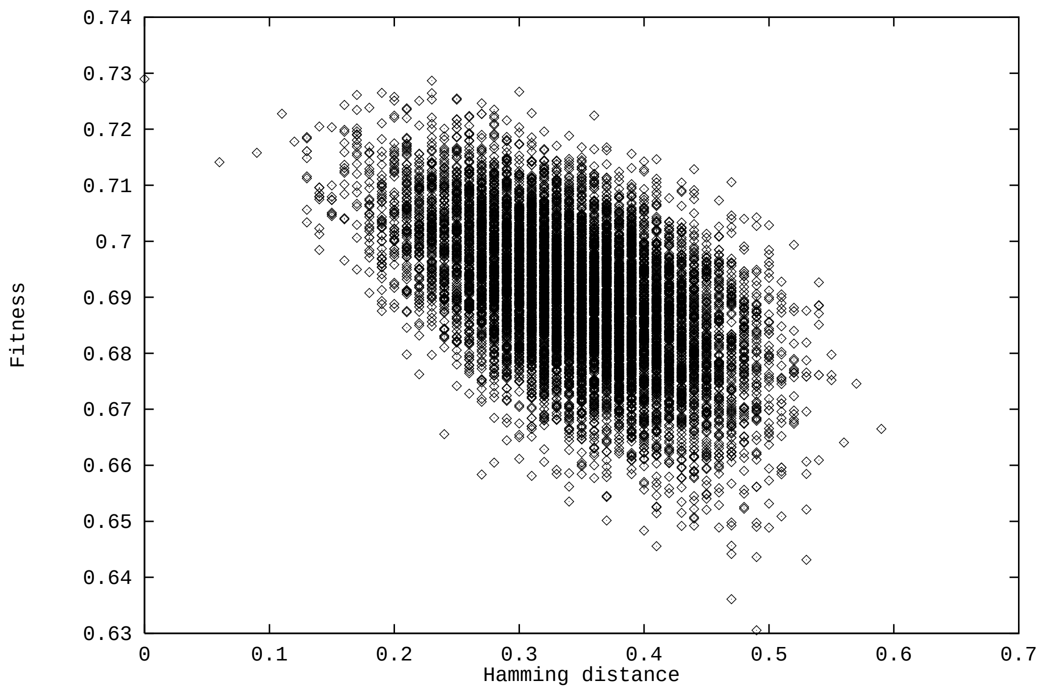

Figure 6.10: The correlation between the fitness of local optima and their (normalized) Hamming distance from the fittest local optimum found. K=2, nearest neighbor interactions.

6.3 Conclusions

There appears to be a clear relation between on the one hand the type of recombination that is used and the type and amount of epistatic interactions on the fitness landscape, and on the other hand the usefulness of recombination.

In the first phase of a search, when a good region in the landscape is searched for by global search, one-point crossover works better when the epistatic interactions are nearby than when they are randomly distributed. In the latter case, one-point crossover is too disruptive. Uniform crossover works best on NK-landscapes with intermediate values of K (K=5,25) and with the interactions randomly distributed.

On a landscape with random interactions, uniform crossover is faster than one-point crossover in finding good regions when the landscape is smooth or rugged (K=0,2,5). On very rugged and completely random landscapes (K=25,50,99), however, one-point crossover is faster than uniform crossover. When the interactions are the nearest neighbors, one-point crossover is the better type in the first phase of the search. As expected, uniform crossover is too disruptive in this case.

Furthermore, there also is clear relation between the locations of local optima in the fitness landscape, and the usefulness of recombination. Recombination is most useful when relatively high optima tend to be near each other. Recombining the information of the location of two optima gives a fair chance of finding even better optima. When the highest optima are not really close enough to each other, recombination becomes less useful.

With these conclusions, the validity of the next statement made by Kauffman can be assessed: "*recombination is useless on uncorrelated landscapes but useful under two conditions: (1) when the high peaks are near one another and hence carry mutual information about their joint locations in the fitness landscape and (2) when parts of the evolving individuals are quasi-independent of one another and hence can be interchanged with modest chances that the recombined individual has the advantage of both parents*". The second condition means that the epistatic interactions should be the nearest neighbors, and not randomly distributed.

That recombination is useless on uncorrelated landscapes is already validated in Chapter 5. The first condition is also validated, considering the conclusion above, based on the results of Section 6.2. The second condition, however, is not validated. As the results in Section 6.1 show, the usefulness of recombination depends on the type of recombination that is used and the type and amount of epistatic interactions on the landscape. It is not always necessary that the "parts of the evolving system are quasi-independent of one another". So, the results presented here do not fully support Kauffman's statement about the usefulness of recombination, but imply a more extensive one.

From these conclusion, some implications can be derived again for both problem solving and biology. From the viewpoint of problem solving, the clear relation between the type of recombination used and the type and amount of epistatic interactions, and the usefulness of recombination, is very important. Knowing the type and amount of epistatic interactions on a landscape makes it possible to choose the type of recombination that is best for this of landscape. "Best" in a sense that it is able to find the best regions, that is containing the highest peaks, or that it finds such good regions faster than other types of recombination.

From a biological viewpoint, something can be said again about the type of landscapes on which more complex organisms, using sexual reproduction, evolve. It appears that the epistatic interactions in the genetic material (the chromosomes) of organisms are randomly distributed. Furthermore, the type of recombination that Nature uses, is *n-point crossover*. Multiple crossover points are randomly selected, and the genetic material between every next pair of crossover points is exchanged. So, this type of recombination is somewhere in between one-point crossover and uniform crossover, in terms of ability to combine building blocks, and in terms of disruptiveness.

For random interactions, uniform crossover works better for low epistasis (K=0, 2, 5), and one-point crossover works better for intermediate to high epistasis (K=25, 50). Since n-point crossover is somewhere between one-point and uniform crossover, it will probably work best for low to intermediate epistasis (K=5 to 25) when the interactions are randomly distributed. In Chapter 5 it was already argued that the amount of epistatic interactions is probably relatively small, compared with the length of the genotypes (in the order of $0.05N$ for example, N being the length of the genotype). So, the conclusions drawn in this chapter seem to agree rather well with this argument.

Chapter 7

Conclusions and Further Research

The goal of this thesis has been to gain more insight into the population flow on fitness landscapes, which hopefully contributes to a theory relating the structure of fitness landscapes to the flow of a population on it. Such a theory can help both in biology, for a better understanding of evolution, and in problem solving, for finding better ways to solve problems by evolutionary search strategies.

To reach this goal, a procedure to determine and express the correlation structure of fitness landscapes has been proposed and applied first (see Chapter 4). Then, different search strategies were applied to different fitness landscapes, to gain some insight into the population flow in general. Besides, the validity of Kauffman's statement about three time scales in adaptation was assessed (see Chapter 5). Finally, the usefulness of recombination was examined more thoroughly, and the validity of Kauffman's statement about this usefulness was assessed (see Chapter 6).

To conclude this thesis, the major conclusions reached in the previous chapters are summarized in this chapter. At the end, some directions for further research are given.

7.1 The structure of fitness landscapes

The structure of a fitness landscape incorporates many features, some of which are local, others are global features. One way to denote the global structure of a fitness landscape is by its correlation structure. The correlation structure of a fitness landscape is determined by the amount of correlation between the fitness of neighboring points in the landscape. In Chapter 4 it is found that this correlation structure can be expressed by an AR(1) model, which has the form

$$y_t = c + \alpha_1 y_{t-1} + \varepsilon_t$$

This AR(1) model is obtained by applying a time series analysis, the Box-Jenkins approach, to a time series of fitness values obtained by generating a random walk with a genetic operator that visits neighboring points in the fitness landscape. Every fitness landscape has its own specific values for the parameters of this model, which are estimated in the time series analysis.

The value of α_1 gives the correlation coefficient between the fitness of two points one step apart. The parameter ε_t is a stochastic variable, and its variance, also estimated in the time series analysis, indicates its amount of influence in the model. The R^2, a measure of goodness of fit of the estimated model, indicates the explanatory (apart from the stochastic component) and predictive value of the model. Furthermore, the fact that a random walk can be modelled by an AR(1) model, means that the fitness of the current point totally depends on the fitness of one step ago. Knowing the fitness of the point two steps ago gives no extra information about the expected fitness of the current point.

Random walks can also be generated with other genetic operators, which do not necessarily visit neighboring points in the landscape. The models obtained from a time series analysis of the data generated this way, indicate how other operators experience the correlation structure of a particular landscape. So, for every operator, a model can be determined and estimated on every type of landscape. The different landscapes and the performance of the different operators can then be compared in terms of these models.

7.2 Time scales in adaptation

Kauffman identifies three natural time scales in adaptation on rugged fitness landscapes:

1. Initially, fitter individuals are found faster by long jumps than by local search. However, the waiting time to find such fitter individuals doubles each time one is found.

2. Therefore, in the midterm, adaptation finds nearby fitter individuals faster than distant fitter individuals and hence climbs a local hill in the landscape. But the rate of finding fitter nearby individuals first dwindles and then stops as a local optimum is reached.

3. On the longer time scale, the process, before it can proceed, must await a successful long jump to a better hillside some distance away.

The results of Chapter 5, however, show that the validity of the first time scale is highly dependent on the type of local search that is used. Instead of these three time scales, the results imply a general picture of only two time scales, or phases, in adaptation:

1. Initially, a relatively good region in the landscape is found by global search.

2. On the longer time scale, local search has to "fine tune" the adapting population by finding the best peaks within this relatively good region.

This holds, of course, provided that selection is able to hold the population within this relatively good region. But this sustained fitness requirement, on the other hand, limits the evolvability of the population. It is still unclear whether both evolvability and sustained fitness can be jointly optimized, but the results in Chapter 5 seem to give a negative answer to this question.

7.3 The usefulness of recombination

Kauffman makes the next statement about the usefulness of recombination: "*recombination is useless on uncorrelated landscapes but useful under two conditions: (1) when the high peaks are near one another and hence carry mutual information about their joint locations in the fitness landscape and (2) when parts of the evolving individuals are quasi-independent of one another and hence can be interchanged with modest chances that the recombined individual has the advantage of both parents*".

The results in Chapter 6 indicate that this statement is only partially correct. Indeed, recombination is useless on uncorrelated landscapes. Also, the first condition that the high peaks in the landscape have to be near one another, must be met, as is shown in Chapter 6. The second condition, however, is too restricted.

There is a clear relation between on the one hand the type of recombination that is used and the type and amount of epistatic interactions on the fitness landscape, and on the other hand the usefulness of recombination. So, it is not necessary that the epistatic interactions are nearby and not randomly distributed, but depending on the type and amount of epistatic interactions on the landscape, a type of recombination can be chosen that works well on such a landscape.

7.4 Directions for further research

All experiments presented in this thesis, were done on "static" landscapes, i.e. the landscape does not change during the search. In problem solving this will be the case most of the time, but from a biological point of view this is only partially plausible. In Nature, landscapes change all the time because the environment changes. So, it would be more plausible to incorporate this in the landscape models. In fact, Kauffman already did this with his coupled NK-landscapes. It would be interesting to look at the population flow on these coupled landscapes, and how, or if, the correlation structure of these landscapes changes during adaptation.

Furthermore, the most plausible model of an adapting population, also from a biological point of view, is the Genetic Algorithm. Only one specific GA is used here, and it would be very interesting, also from the point of view of problem solving, to do some of the experiments with different parameters for the GA. This could give some more insight into the sustained fitness versus evolvability problem.

Trying other hybrid strategies might also yield interesting results. For example combining the GA with Long jumps, when the population becomes stuck in one particular region of the landscape. Or maybe still other recombination methods exist that appear to be useful in combination with some sort of epistatic interactions on the landscape.

A lot still has to be done to find a real theory relating the structure of rugged multipeaked fitness landscapes to the flow of a population upon those landscapes. But hopefully the research presented in this thesis will shed some light on what such a theory might look like.

— Let There Be More Light —

Appendix A

A Two-Sample Test for Means

Suppose $X_i,\ i = 1..n_1$ and $Y_j,\ j = 1..n_2$ are the observed values of (independent) random samples from two probability distributions with mean μ_1 and μ_2 respectively. Then the following hypothesis can be tested:

$$H_0 : \mu_1 = \mu_2$$

This hypothesis is rejected with $100(1-\alpha)\%$ confidence if

$$|t_0| \geq t_{1-\alpha/2}(v)$$

where α is the significance level (the probability of rejecting a true hypothesis), and $t(v)$ is a Student's t distribution with v degrees of freedom. The values of t_0 and v are calculated as follows:

$$t_0 = \frac{\overline{X} - \overline{Y}}{\sqrt{S_1^2/n_1 + S_2^2/n_2}}$$

$$v = \frac{(S_1^2/n_1 + S_2^2/n_2)^2}{[(S_1^2/n_1)^2/(n_1-1)] + [(S_2^2/n_2)^2/(n_2-1)]}$$

where

$$\begin{aligned}
\overline{X} &= \frac{1}{n_1}\sum_{i=1}^{n_1} X_i \\
\overline{Y} &= \frac{1}{n_2}\sum_{j=1}^{n_2} Y_j \\
S_1^2 &= \frac{\sum_{i=1}^{n_1}(X_i - \overline{X})^2}{n_1 - 1} \\
S_2^2 &= \frac{\sum_{j=1}^{n_2}(Y_j - \overline{Y})^2}{n_2 - 1}
\end{aligned}$$

Appendix B

The Height of Peaks in a Landscape

It appears that the fitness of the highest peaks in NK-landscapes for low values of K (K=2, 5) is higher than the fitness of the peak in a landscape with K=0. If K increases more (K=25, 50), then the fitness of the peaks gradually decreases, and eventually becomes less than that of the peak for K=0. This appendix gives an explanation for this phenomenon.

In the NK-model, the fitness of a bit depends on its own value and the value of K other bits. So there are 2^{K+1} possible "configurations" for a bit, all of which are assigned a random fitness from a UNIF(0,1) distribution. The higher the value of K, the higher the chance that some of the configurations of a bit are assigned a high fitness, because more drawings from the same distribution are done.

The expected values of the fitnesses that are assigned to the configurations of a bit can be calculated by means of *order statistics*. Suppose $x_1, x_2, ..., x_n$ is a random sample of size n from some continuous *probability density function*[1] (pdf) $f(x)$, $a \leq x \leq b$. When this random sample is ordered in increasing order, a set of order statistics, denoted by $y_1, y_2, ...y_n$, is obtained. So y_1 is the minimum of $\{x_1, x_2, ..., x_n\}$ and y_n is the maximum of this set of observed values.

The pdf of the r^{th} order statistic y_r is defined as:

$$g_r(y_r) = \frac{n!}{(r-1)!(n-r)!}[F(y_r)]^{r-1}[1-F(y_r)]^{n-r}f(y_i), \quad a \leq y_r \leq b$$

where $F(x)$ is the *cumulative distribution function*[2] (CDF) belonging to the pdf $f(x)$. The expected value of this r^{th} order statistic y_r can now be calculated as:

$$E[y_r] = \int_a^b y g_r(y) dy$$

[1] A function $f(x)$ is a pdf for some continuous random variable X if and only if $f(x) \geq 0$ $\forall x$ and $\int_{-\infty}^{\infty} f(x)dx = 1$.

[2] The CDF $F(x)$ denotes the chance that a random variable X will be less than or equal to the value x. It is defined as $F(x) = P[X \leq x] = \int_{-\infty}^{x} f(t)dt$, where $f(x)$ is the pdf of the random variable X.

In the NK-model, for every bit 2^{K+1} drawings from a UNIF(0,1) distribution are done. So $n = 2^{K+1}$, $f(x) = 1$, and $F(x) = x$, $0 \leq x \leq 1$. The pdf of the r^{th} order statistic is then

$$\begin{aligned}
g_r(y_r) &= \frac{n!}{(r-1)!(n-r)!} y_r^{r-1}(1-y_r)^{n-r} \cdot 1 \\
&= \frac{n!}{(r-1)!(n-r)!} y_r^{r-1} \sum_{i=0}^{n-r} \binom{n-r}{i} (-1)^i y_r^i \\
&= \frac{n!}{(r-1)!(n-r)!} \sum_{i=0}^{n-r} \binom{n-r}{i} (-1)^i y_r^{i+r-1}
\end{aligned}$$

and the expected value of the r^{th} order statistic becomes

$$\begin{aligned}
E[y_r] &= \int_0^1 \frac{n!y}{(r-1)!(n-r)!} \sum_{i=0}^{n-r} \binom{n-r}{i} (-1)^i y^{i+r-1} dy \\
&= \int_0^1 \frac{n!}{(r-1)!(n-r)!} \sum_{i=0}^{n-r} \binom{n-r}{i} (-1)^i y^{i+r} dy \\
&= \left[\frac{n!}{(r-1)!(n-r)!} \sum_{i=0}^{n-r} \frac{1}{i+r+1} \binom{n-r}{i} (-1)^i y^{i+r+1} \right]_0^1 \\
&= \frac{n!}{(r-1)!(n-r)!} \sum_{i=0}^{n-r} \binom{n-r}{i} \frac{(-1)^i}{i+r+1}
\end{aligned}$$

From this result the expected value of the highest order statistic can be found by substituting n for r:

$$\begin{aligned}
E[y_n] &= \frac{n!}{(n-1)!(n-n)!} \sum_{i=0}^{n-n} \binom{n-n}{i} \frac{(-1)^i}{i+n+1} \\
&= n \times \sum_{i=0}^{0} \binom{0}{i} \frac{(-1)^i}{i+n+1} \\
&= n \times \binom{0}{0} \frac{1}{n+1} \\
&= \frac{n}{n+1}
\end{aligned}$$

When K=0 in the NK-model, the (global) optimum can be found by taking for every bit the value which is assigned the highest fitness (see Section 2.3). The expected value of this fitness is $E[y_n] = \frac{n}{n+1} = \frac{2}{3}$ ($n = 2^{0+1} = 2$). But when $K > 0$, the optimum can not be found in such a way because of the conflicting constraints that result from the epistatic interactions. The expected value of the maximum of all the fitnesses assigned to the configurations of a bit, increases with increasing K, but the conflicting constraints also become more stringent, so a fitness somewhere further back in the order statistics will be the best possible for most bits.

Table B.1 gives the expected values of the order statistics y_r for K=0,2 and 5. The average fitness of the highest optimum found with hillclimbing for these landscapes is 0.67046, 0.73001 and 0.75030 respectively. So, indeed the fitness of the optimum for K=0 is equal to the expected fitness of the highest order statistic (as explained above). The fitness of the highest optimum for K=2 appears to be somewhere between the expected fitness of the second and third highest order statistic, and the fitness of the highest optimum for K=5 is somewhere between the expected value of the 48th and 49th order statistic from a total of 64.

r	K=0	K=2	K=5
1	0.33333	0.11111	0.01538
2	0.66666	0.22222	0.03077
3		0.33333	0.04615
4		0.44444	0.06154
5		0.55555	0.07692
6		0.66666	0.09231
7		0.77777	0.10769
8		0.88888	0.12308
9			0.13846
10			0.15385
$\vdots$			$\vdots$
48			0.73846
49			0.75385
$\vdots$			$\vdots$
63			0.96923
64			0.98462

Table B.1: The expected values of the order statistics of the fitnesses assigned to the possible configurations of a bit for different values of K.

So, this table shows that the higher the value of K, the further back in the list of expected values of the order statistics of the fitness values assigned to the 2^{K+1} possible configurations of a bit one has to go to find the maximum possible value, considering the conflicting constraints. For low values of K, this maximum possible value is still higher than that for K=0. But eventually, when K increases further, the conflicting constraints become too stringent, and a fitness much further back in the list of order statistics is the best possible, being a lower fitness than the maximum for K=0.

Appendix C

Used Software

A great deal of the software that was used for the experiments was written by myself. This includes:

- A fitness function for the NK-model.
- A program for generating random walks on a fitness landscape.
- Two hillclimbing programs.
- A program for finding and storing different optima in a landscape.

All this was written in C++, running under both UNIX (SunOS 4.1.3) and MS-DOS.

The fitness function for the NK-model was tested by repeating an experiment, done originally by Kauffman, for finding the mean fitness of local optima and the mean walk lengths to these local optima in NK-landscapes ([Kau93], pages 55-57). The obtained results were very similar to those in [Kau93], indicating that the fitness function is implemented correctly.

Rémon Sinnema wrote a very nice toolkit for working with GA's, called EUREGA (also in C++). I added different kinds of operators to this toolkit for performing the experiments with the different search strategies.

The Box-Jenkins approach (Chapter 4) was carried out with the statistical package TSP (Time Series Processor). For the two-sample test for means (Appendix A), both a spreadsheet (PlanPerfect) and the statistical package SPSS were used. The order statistics in Appendix B were calculated with Maple. The graphs in this thesis were produced with GNUPLOT. For those of you who did not recognize it already, the thesis itself was written in LaTeX.

Bibliography

[BE87] L. J. Bain and M. Engelhardt. *Introduction to Probability and Mathematical Statistics.* Duxbury Press, 1987.

[BHS91] T. Bäck, F. Hoffmeister, and H-P. Schwefel. A Survey of Evolution Strategies. In R. K. Belew and L. B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 2–9. Morgan Kaufmann, 1991.

[BJ70] G. E. P. Box and G. M. Jenkins. *Time Series Analysis, Forecasting and Control.* Holden Day, 1970.

[Dar59] C. Darwin. *The Origin of Species by Means of Natural Selection.* Penguin Books, 1859.

[Dav91] L. Davis, editor. *Handbook of Genetic Algorithms.* Van Nostrand Reinhold, 1991.

[EJ89] M. A. Edey and D. C. Johanson. *Blueprints—solving the mystery of evolution.* Little, Brown and Company, 1989.

[ERR94] A. E. Eiben, P-E. Raué, and Zs. Ruttkay. Repairing, adding constraints and learning as a means of improving GA performance on CSPs. In J. C. Bioch and S. H. Nienhuys-Cheng, editors, *Proceeding of the Fourth Belgian-Dutch Conference on Machine Learning*, pages 112–123, 1994.

[FM] S. Forrest and M. Mitchell. What Makes a Problem Hard for a Genetic Algorithm? Some Anomalous Results and Their Explanation.

[FM93] S. Forrest and M. Mitchell. Relative Building-Block Fitness and the Building-Block Hypothesis. In D. Whitley, editor, *Foundations of Genetic Algorithms 2*, pages 109–126. Morgan Kaufmann, 1993.

[Gol89] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, 1989.

[Gra89] C. W. J. Granger. *Forecasting in Business and Economics.* Academic Press, 2nd edition, 1989.

[Gre86] J. J. Grefenstette. Optimization of Control Parameters for Genetic Algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, (1):122–128, 1986.

[Hol92] J. H. Holland. *Adaptation in Natural and Artificial Systems.* MIT Press, 2nd edition, 1992.

[J$^+$88] G. G. Judge et al. *Introduction to the Theory and Practice of Econometrics.* John Wiley & Sons, 2nd edition, 1988.

[Kau89] S. A. Kauffman. Adaptation on Rugged Fitness Landscapes. In D. Stein, editor, *Lectures in the Sciences of Complexity*, pages 527–618. Addison-Wesley, 1989.

[Kau93] S. A. Kauffman. *Origins of Order: Self-Organization and Selection in Evolution.* Oxford University Press, 1993.

[Lip91] M. Lipsitch. Adaptation on Rugged Landscapes Generated by Iterated Local Interactions of Neighboring Genes. In R. K. Belew and L. B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 128–135. Morgan Kaufmann, 1991.

[MHF] M. Mitchell, J. H. Holland, and S. Forrest. When Will a Genetic Algorithm Outperform Hill Climbing?

[MS93] J. Maynard Smith. *The Theory of Evolution.* Cambridge University Press, Canto edition, 1993.

[MWS91] B. Manderick, M. de Weger, and P. Spiessens. The Genetic Algorithm and the Structure of the Fitness Landscape. In R. K. Belew and L. B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 143–150. Morgan Kaufmann, 1991.

[TG94] D. Thierens and D.E. Goldberg. Elitist recombination: an integrated selection recombination GA. In *Proceedings of the IEEE World Gongress on Computational Intelligence*, pages 508–512, 1994.

[Wei90] E. D. Weinberger. Correlated and Uncorrelated Fitness Landscapes and How to Tell the Difference. *Biological Cybernetics*, (63):325–336, 1990.

[WH88] N. K. Wessels and J. L. Hopson. *Biology.* Random House, Inc., 1988.