Department of Information Technology

technical report NUIG-IT-100102

Analysis of the performance of Genetic Algorithms and their Operators using Kauffman's NK Model

S. Hill (NUI, Galway)
C. O' Riordan (NUI, Galway)

Analysis of the performance of Genetic Algorithms and their Operators using Kauffman's NK Model.

Seamus Hill, Colm O' Riordan
Dept. of Information Technology,
NUI, Galway

Abstract

This paper outlines the operators and operations of Genetic Algorithms, and Kauffman's NK model. To analyse the performance of genetic algorithms and their operators the fitness landscape is crucial. Kauffman's NK model is used to carry out analysis of the performance of a genetic algorithm and its crossover and mutation operators. Future work which may extend from this include using the NK model to: analyse the performance of the inversion operator; assist the development of coding schemes; analyse real world problems in order to assist in selecting optimum rates for operators, and studying adaptive rates for operators.

1 Introduction

The book *Adaptation in Natural and Artificial Systems* written by John Holland, presents Genetic Algorithms (GA's) as an abstraction of biological evolution and presents a theoretical framework which can be adopted for GA's [3]. According to Holland, the GA is a method of moving from one population of bit strings which represent creatures or possible solutions to a problem, to a new population, using selection along with crossover, mutation and inversion operators.

Given an optimisation problem, the set of possible solutions to this problem can be encoded using strings of a fixed length formed from some finite sized alphabet. This encoding generates a representation space, which is a high dimensional space of all possible strings of a particular length, each of which encodes a possible solution to the problem. The effect of the choice of operators used and their associated rate has been a topic of much debate and research.

To develop a theory regarding the choice of specific genetic operators (e.g. the inversion operator) to include in a GA and the rate at which they should be set, a model such as Kauffman's NK model can be used to analyse their performance. This is because the structure of a fitness landscape depends on the underlying problem, and developing a theory about operators and their probability rates should be independent of the structure of the landscape. To this end, Kauffman's NK model provides a problem independent landscape, as the fitness landscape can be gradually turned from smooth to rugged.

2 An Overview of Genetic Algorithms

Genetic Algorithms (GAs) are search algorithms based on the mechanics of natural selection and natural genetics. They are a combination of survival of the fittest among string structures, and a structured yet randomised information exchange to form a search algorithm which has intelligent search qualities. With each generation, a new set of artificial creatures (strings) are created using parts of the fittest of the old; combined with some new parts. Although randomised, genetic algorithms efficiently exploit historical information to speculate simple search points with expected improved performance. GA's can be classified as a group of computational models inspired by natural evolution. A potential solution to a particular problem is encoded on a data structure similar to a chromosome. Each chromosome consists of a number of "genes" (e.g. bits), with each gene being an instance of a particular "allele" (e.g., 0 or 1). Recombination operators are applied to this binary string so as to preserve critical information. A GA commences with a population of typically randomly generated chromosomes. The selection operator chooses which chromosomes in the population will be allowed to reproduce. The chromosomes are allocated reproductive opportunities in a way that chromosomes which represent a better solution to the target population are given a higher

chance to "reproduce" than those chromosomes which have achieved lower fitness evaluations It also decides how many offspring each chromosome can have with fitter chromosomes having a greater chance to produce more offspring. The fitness of a solution is typically defined with respect to the current population.

Following selection the crossover operator exchanges sub-parts of two chromosomes, normally with a probability rate typically between 85% and 90%, this can be compared to biological recombination between two single-chromosome organisms and can be viewed as sexual recombination found in nature. The crossover operator randomly chooses a locus and exchanges the parts of the strings before and after that locus between two chromosomes to create two offspring. For example, the strings *10000100* and 11111111 could be crossed over after the third locus in each to produce the two offspring *10*11111 and 11*00100*.

The mutation operator changes randomly the values of a location on the chromosome. It randomly flips some of the bits in a chromosome. For example, the string 00000100 might be mutated in its second position to yield 01000100. Mutation can occur at each bit position in a string with some probability, usually very small (e.g., 0.001) [9].

The order in which genes are arrayed in the chromosome can be rearranged by the inversion operator, which reverses the order of a contiguous section the chromosome [8]. Most methods called "GAs" have the following elements in common: populations of chromosomes, selection according to fitness, crossover to produce new offspring, and random mutation of new offspring.

3 Operator Overview

Reproduction combined with Crossover provide GA's with the bulk of their processing power. Mutation on the other hand plays a secondary role in the operation of genetic algorithms. It is needed because, "even though reproduction and crossover effectively search and recombine extant notions, occasionally they may become overzealous and loose some potentially useful genetic material (1's or 0's at particular locations). In artificial genetic systems, the mutation operator protects against such an irrecoverable loss. By itself, mutation is a random walk through the string space. When used sparingly with reproduction and crossover, it is an insurance policy against premature loss of important notion" [2]. Goldberg also notes that the frequency of mutation to obtain good results in empirical GA studies is "on the order of one mutation per thousand bit (position) transfer".

4 The Fitness Landscape

4.1 Bit strings and Hamming distance

To describe fitness landscapes, a notion of a distance between genotypes is needed. Genotypes are codings, and different codings can cause different distance measures. Another point is that often there are more than one distance measure, or metric, which can be defined for one and the same coding. Usually, a coding in the form of bit strings is used. 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 normalised Hamming distance can be defined by dividing the Hamming distance between two bit strings by the length of these bit strings. By adopting this approach, the distance measure is independent of the length of the bit strings. A normalised Hamming distance of 0.5, for example, means that half the bits of two bits string have a different value [5].

4.2 The Genotype Space

If the possible solutions for a given problem are encoded by some form of genotype, then the problem space can also be represented by a genotype space. A genotype space is the 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 i.e. the Hamming distance). All the points at distance one are called the neighbours of this first point, and together they form a neighbourhood, i.e. 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 as metric, every bit string of length three has exactly three neighbours, namely those bit strings that differ in one of the three bits. Every point in the space represents one genotype and has exactly three neighbours, each of which differs in the value of one bit [5].

4.3 The Fitness Landscape

The idea of a fitness landscape is used as a framework for thinking about evolution. Biological organisms can be characterised by their genotype, which is the genetic "encoding" of the organism, or their phenotype, which is

the actual form and behaviour of the organism. A fitness value can be assigned to each phenotype, which denotes its ability to survive and reproduce. Evolution can be viewed as a process that searches, by means of genetic operators like crossover and mutation, a fitness landscape of possible genotypes, looking for genotypes that encode highly fit phenotypes. Put another way, evolution searches for solutions, encoded in genotypes, for the problem of finding fit organisms that are capable of reproduction. Every genotype will have a relative fitness assigned to it. This is determined by a fitness function. The fitness landscape is then constructed by assigning the fitness values of the genotypes to the corresponding points in the genotype space. To visualise this picture consider each point in the genotype space being given a "height" according to its fitness. From this a "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 neighbours [4].

To summarise, the structure of a landscape incorporates many things, such as the number of neighbours 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 neighbouring points is relatively small, is called smooth. On such a landscape it will be easy to find good optima as local information about the landscape can be used effectively to direct the search. A landscape with a relatively large average fitness difference between neighbours, is called rugged. On such a landscape it will be difficult to find good optima, in other words local information becomes less valuable. So, the global structure of a landscape can range from very smooth to very rugged. 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 [5].

5 NK Model

5.1 Description of the NK Fitness Model

> "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

Stuart Kauffman devised the "NK fitness Landscape" model to explore the way that epistasis controls the "ruggedness" of an adaptive landscape. He wanted to specify a family of fitness functions whose ruggedness could be 'tuned' by a single parameter. This was achieved by building up multiple 'atoms' of maximal epistasis. The NK model is a stochastic method for generating a fitness function F:$\{0,1\}^N \rightarrow \mathcal{R}^+$ on binary strings $x \in \{0,1\}^N$, where the genotype x consists of N loci, with two possible alleles at each locus x_i. It has two basic components: a structure for gene interactions, and a way this structure is used to generate a fitness function for all possible genotypes. The gene interaction structure is created as follows: the genotype's fitness is the average of N fitness components contributed by each locus. Each gene's fitness component F_i is determined by its own allele, x_i, and also the alleles at K other epistatic loci (therefore K must fall between 0 and N - 1). These K other loci could be chosen in any number of ways from the N loci in the genotype. Kauffman investigated two different alternatives: *adjacent neighbourhoods*, where the K genes nearest to locus i on the chromosome are chosen; and *random neighbourhoods*, where these K other loci are chosen randomly on the chromosome. In the adjacent neighbourhood model, the chromosome is taken to have periodic boundaries, so that the neighbourhood wraps around the other end when it is near the terminus [1].

Epistasis is implemented through a "House of Cards" model of fitness effects, in other words, whenever an allele is changed at one locus, all of the fitness components with which the locus interacts are changed, without any correlation to their previous values. Thus a mutation in any of the genes affecting a particular fitness component is like pulling a card out of a house of cards - it tumbles down and must be rebuilt from scratch, with no information passed on from the previous value. Kauffman implemented this by generating, for each fitness component, a table of 2^{K+1} numbers for each allelic combination for the (K+1) loci determining that fitness component. These numbers are independently sampled from a uniform distribution on [0,1]. The consequence of this individual resampling of fitness components is that the fitness function develops conflicting constraints: a mutation at one gene may improve its own fitness component, but decreases the fitness component of another gene with which it interacts. Also, if the allele at another interacting locus changes, an allele that had been optimal may no longer be optimal. Therefore, epistatis interactions provide "frustration" in trying to optimise all genes simultaneously [1].

The NK model was introduced by Kauffman to have a problem independent model for constructing fitness landscapes that can gradually be tuned from smooth to rugged. The main parameters of the model are N, the number of

genes in the genotype, i.e. the length of the strings that form the points in the landscape, and K, the number of other genes that epistatically influence a particular gene (i.e., the fitness contribution of each gene is determined by the gene itself plus K other genes) [5]. K sets the level of epistasis by determining the dependence the partial fitness of a gene at location n has on the genes in a neighbourhood of K other locations. The neighbourhood may be at the K locations nearest to n in the genotype or a set of K locations randomly picked from anywhere on the genotype. Following this a series of N lookup tables are then generated, one for each gene location in the genotype. Each table has 2^{K+1} random entries in the interval (0,1). The fitness, F_{NK}, of a particular genotype is calculated by the function:

$$F_{NK} = \frac{1}{N} \sum_{n=1}^{N} f(x)$$

where the partial fitness f(n) is obtained from the nth lookup table using the values of the genes in location n and its neighbourhood as the lookup key [7]. Below is an example to illustrated the calculation of $f(n)$ with N=8, K=2. In this example n = `011`, when the table is refered to $f(n)$ is shown to be `0.432809`.

```
Neighbourhood of n
      |       |
```

Genotype:

1	0	0	0	1	1	0	0

nth Lookup Table

0	0	0	0.724367
0	0	1	0.123989
0	1	0	0.987432
0	1	1	0.432809
1	0	0	0.987234
1	1	0	0.349566
1	1	0	0.274095
1	1	1	0.521926

6 Experiment Design

To attempt to analyse the performance of a genetic algorithm using Kauffman's NK model, it was decided to use the simple genetic algorithm as outlined by Goldberg [2][1].

[1]A C-language Implementation of a Simple Genetic Algorithm has been written by Robert E. Smith (The University of Alabama), David E. Goldberg (The University of Illinois) and Jeff A. Earickson (Alabama Supercomputer Network).

6.1 Simple Genetic Algorithm (SGA)

The program is C-language translation and extension of the original Pascal SGA code presented by Goldberg. The SGA-C is intended to be a simple program for first-time GA experimentation. It is not intended to be definitive in terms of its efficiency or the grace of its implementation.

6.2 The NK Model

The implementation[2] of the NK model used is limited to NK landscapes of N=32 and K=31 or smaller. If the preload option in initModel is true, the entire landscape will be constructed during the initialization phase. A preload option of false will cause the landscape to be constructed on the fly. Constructing the landscape on the fly is necessary for large values of n and k due to the memory required to store the entire landscape. The model constructs random tables, which grow exponentially as K increases. That is 2^{K+1} rows and N columns. The evalString routine evaluates the fitness of a binary string chromosome of length 32 or smaller. A value between 0 and 1 will be returned.

This NK-landscape problem generator models the idea presented by Stuart A. Kauffman [6]. The user inputs a binary chromosome of length N. Each gene makes a fitness contribution based on its allele (1 or 0) and the allele of K other genes. The fitness contributions are drawn from a uniform distribution ranging from 0.0 to 1.0. The fitness of a chromosome is the average fitness of the genes at all N loci.

The landscape characteristics are as follows; where K=0 there is a single peak and the fitnes of strings is highly correlated with Hamming distance. Where K=N-1 there are many sub-optimal peaks and the fitness of strings is uncorrelated with Hamming distance. As both K and N increse, an increasing number of fitness peaks fall towards the mean fitness as a result of conflicting constraints among the genes.

7 Experiment Execution

To conduct the experiment the fitness function of the SGA-C was replaced by the NK-Landscape generator. This allows for a problem independent method of testing the performance of the SGA-C over various landscapes.

[2]A C language implementation of Kauffman's NK model called the NK-landscape problem generator has been written by Mitchell A. Potter, and is available at http://www.cs.gmu.edu/ mpotter/nk-generator/

By allowing various rates of crossover and various rates of mutation to be passed to the genetic algorithm, it is possible to plot the fitness rates for each generation at each particular rate of crossover and each rate of mutation for the various levels of complexity which ranged from 0 to 31. The experiment was conducted in two parts, the first, was to discover the fitness levels for various crossover rates. The second part, involved discovering the fitness rates for different mutation rates. Both parts also included landscape altertation.

During the experiment the GA was run 5 times with a population size of 200 for 200 generations and a fixed length chromosome of 32 and a fixed random seed. Also for the first set of experiments the level of mutation was fixed at 0.001%, while for the second set of experiments the level of crossover was fixed at 80%. In first experiment the rate of crossover varied fron 0.00 to 100% with 5% increments. Druing the second experiment the level of mutation was increased from 0.001% to 0.511% at intervals of 0.01%.

8 Results

8.1 Crossover

Results of the first experiment with varying rates of crossover indicate that on average, higher rates of fitness are obtained on less complex landscapes. This is illustrated in figure 1 which plots the average best fitness for each landscape with crossover fixed at 90%.

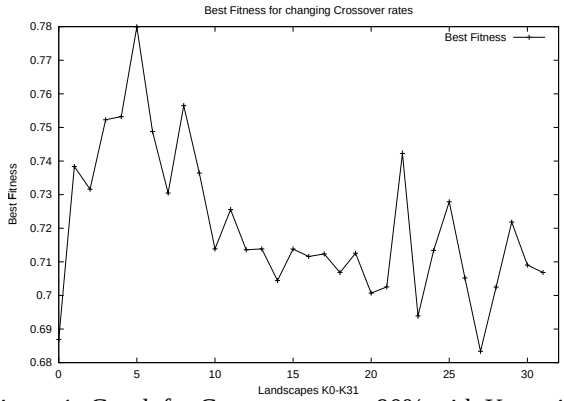

Figure 1: Graph for Crossover set to 90% with K ranging from 0-31

The results also show that as the rates of crossover increase so do the fitness levels (see figure 2 with K set to 0, and figure 3 with K set to 31). In other words, statistically higher fitness levels were achieved at the higher rates of crossover, that is between 85% and 95%. The number of generations also has an effect on the fitness levels as the rates of crossover change. Increases in the best fitness levels appear as generation numbers increase. However, in these experiments 200 generations appear satisfactory as increases in the number of generations provided little increase in fitness levels.

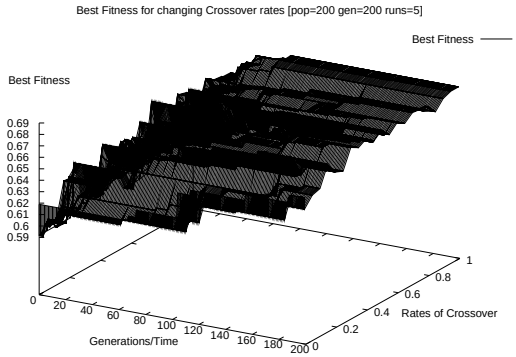

Figure 2: Graph for K set to 0 and N set to 32

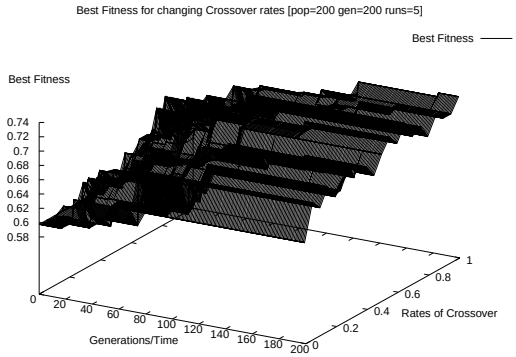

Figure 3: Graph for K set to 31 and N set to 32

8.2 Mutation

In the second experiment where the rates of mutation are altered, results show that increasing rates of mutation are not associated with higher fitness levels. (See figures 4 and 5). The number of generations and the crossover rate has a greater effect when combined with mutation on the fitness levels. Therefore, it seems, illustrating that lower levels of mutation are as effective, but have the additional advantages of being computationally less expensive as well

as reducing the risk of destroying existing fitter schemas.

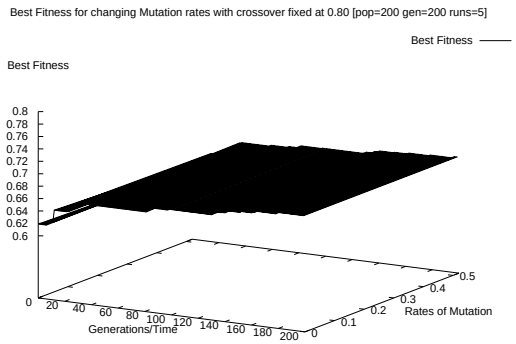

Figure 4: Graph for K set to 0 and N set to 32

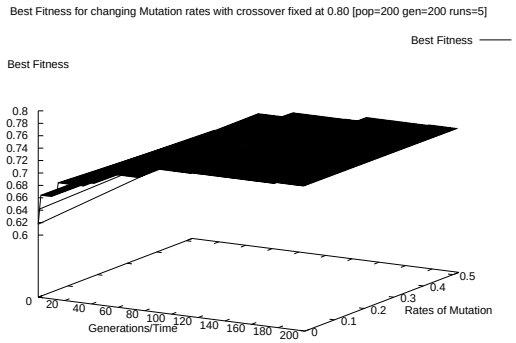

Figure 5: Graph for K set to 31 and N set to 32

It is interesting to note that at lower levels of generations mutation rates appear to have more influence on the best fitness levels, but as the number of generations increase, i.e. 100 generations plus, the difference in best fitness levels are minor.

The results also indicate that the highest average best fitness rates were obtained with lower mutation rates, with the mutation rate of 0.001% achieving the highest (figure 6), illustrating that there is little benefit, if any, in increasing the rates of mutation above the lower rates i.e. 0.001% and 0.010.

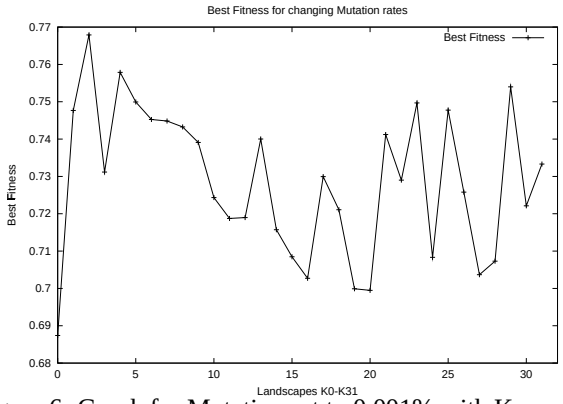

Figure 6: Graph for Mutation set to 0.001% with K ranging from from 0-31

9 Conclusion and Future work

To conclude the results of the experiments appear to indicate that the SGA algorithm and its operators function properly at high rates of crossover and low rates of mutation.

Future work includes conducting more experiments on crossover and mutation using rates which have returned the best results from experiments to date. These would include crossover experiments with a number of other muataion rates such as 0.002%, 0.005% and 0.008%, and mutation experiments with crossover rates of 85%, 90% and 95%. By conducting these experiments the best range for crossover and mutation rates over the full range of landscapes (K from 0 to 31) may converge on the theoretical optimum rates, that is crossover of 85% to 90% and mutation of 0.001%, thereby indicating the successfulness of the genetic algorithm being analysed.

Additional experiments may include the following:

- the addition of an inversion operator to the algorithm and studying its effects;
- mapping new problems, when developing coding schemes attention may be paid to the degree of epistasis amongst genes (and sets of genes) which will allow a successful coding scheme to be developed i.e one that conforms with Goldbergs *Building Block Hypothesis*, i.e. Goldberg [2] argues that the power of a GA lies in being able to find good *building blocks*. These building blocks are schemata of short defining length consisting of bits that work well together, and tend to improve performance when incorporated into a creature. A successful coding scheme encourages building

blocks to form by ensuring that, (I) related genes are close together on the chromosome, while (II) there is little interaction between genes.

- research into the mapping of real world problems to landscapes with a given complexity/difficulty (i.e known measure of epistasis) to assist in selecting the optimum rates for various operators for particular types of problems;
- studying adaptive rates, for example if the position on a landscape determines the mutation rate then the algorithm may be designed to use varying rates of mutation as the landscape changes, and may also indicate a stoping condition.

10 Acknowledgement

We would like to thank Niall Wilson (National Center for Biomedical Engineering Science) for allowing us the use of their High Performance Computing facilities.

References

[1] L. Altenberg. Nk fitness landscapes. In et al.(Eds.) T.Back, editor, *The Handbook of Evolutionary Computation, Section B.2.7.2*. Oxford University Press, 1997.

[2] David E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Publishing Company, Inc., Reading, MA, 1989.

[3] John H. Holland. *Adaptation in natural artificial systems*. University of Michigan Press, Ann Arbor, 1975.

[4] W. Hordijk. A measure of landscapes. Technical Report Technical report SFI Preprint 95-05-049, Santa Fe Institute., 1995.

[5] Wim Hordijk. Population flow on fitness landscapes. Master's thesis, Erasmus University, Rotterdam, 1994.

[6] S.A. Kauffman. Origins of order. Oxford University Press, Oxford, 1993., 1993.

[7] Giles Mayley. The evolutionary cost of learning. In Pattie Maes, Maja J. Mataric, Jean-Arcady Meyer, Jordan Pollack, and Stewart W. Wilson, editors, *From Animals to Animats 4: Proc. of the Fourth Int. Conf. on Simulation of Adaptive Behavior*, Cambridge, MA, 1996. The MIT Press.

[8] M. Mitchell and S. Forrest. Genetic algorithms and artificial life. Artificial Life, 1(3), 1994.

[9] Melanie Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, 1996.