modelling of directed evolution: Implications for experimental design and stepwise evolution

David C. Wedge, William Rowe, Douglas B. Kell, Joshua Knowles

www.elsevier.com/locate/yjtbi

In silico modelling of directed evolution: Implications for experimental design and stepwise evolution

David C. Wedge, William Rowe, Douglas B. Kell, Joshua Knowles

PII: S0022-5193(08)00584-5
DOI: doi:10.1016/j.jtbi.2008.11.005
Reference: YJTBI5360

To appear in: *Journal of Theoretical Biology*

Received date: 8 August 2008
Revised date: 3 November 2008
Accepted date: 3 November 2008

Cite this article as: David C. Wedge, William Rowe, Douglas B. Kell and Joshua Knowles, *In silico* modelling of directed evolution: Implications for experimental design and stepwise evolution, *Journal of Theoretical Biology* (2008), doi:10.1016/j.jtbi.2008.11.005

In silico Modelling of Directed Evolution: Implications for Experimental Design and Stepwise Evolution

David C. Wedge[a,*], William Rowe [a], Douglas B. Kell [a,], Joshua Knowles [a,b]

[a] Manchester Interdisciplinary Biocentre, University of Manchester, 131 Princess Street, Manchester, M1 7ND, U.K.
[b] School of Computer Science, University of Manchester, Manchester, M13 9PL, U.K.

*Corresponding author. Tel.: +44 161 3065145

E-mail addresses: @manchester.ac.uk

Abstract

We model the process of directed evolution (DE) *in silico* using genetic algorithms. Making use of the NK fitness landscape model, we analyse the effects of mutation rate, crossover and selection pressure on the performance of DE. A range of values of K, the epistatic interaction of the landscape, are considered, and high-throughput and low-throughput modes of evolution are compared. Our findings suggest that for runs of or around ten generations' duration - as is typical in DE - there is little difference between the way in which DE needs to be configured in the high- and low-throughput regimes, nor across different degrees of landscape epistasis. In all cases, a high selection pressure (but not an extreme one) combined with a moderately high mutation rate works best, while crossover provides some benefit but only on the less rugged landscapes. These genetic algorithms were also compared with a "model-based approach" from the literature, which uses sequential fixing of the problem parameters based on fitting a linear model. Overall, we find that purely evolutionary techniques fare better than do model-based approaches across all but the smoothest landscapes.

Keywords: genetic algorithm, fitness landscape, NK-landscape, selection pressure, mutation rate

1 Introduction

Directed Evolution (DE) has in recent years emerged as an effective technique for generating and selecting proteins[1] with a variety of uses. The starting point is usually a library containing proteins that already possess the desired function to some extent, although randomly generated proteins have also been used. Through a series of iterative steps, or 'generations', during each of which the proteins are diversified and then screened, the protein library is 'evolved' towards better performance. Proteins have been evolved using DE for a variety of roles including the production of industrial catalysts (Cherry and Fidantsef 2003; Sylvestre *et al.* 2006), thermostable enzymes (Yun *et al.* 2006; Chautard *et al.* 2007), molecule-specific aptamers (Joyce 1994) and vaccines (Piatesi *et al.* 2006).

In this paper we use Genetic Algorithms (GAs) to model the DE process. This seems like a natural thing to do, given the common roots of evolutionary computation and DE. Both are inspired by Darwinian evolution, proceeding through similar steps of selection, reproduction and variation. Further, variation is typically achieved during DE using error-prone polymerase chain reaction (Leung *et al.* 1989) or DNA shuffling (Stemmer 1994; Stemmer 1994; Coco *et al.* 2001). These techniques have similar effects to the mutation and crossover operators commonly used in GAs. It should therefore be possible to simulate the DE process using a GA and an appropriate dataset. However, DE practitioners do not in general apply lessons learned from evolutionary computation in determining experimental parameters, nor *vice versa*. One

[1] DE has been performed on DNA and RNA as well as on proteins. References to 'proteins' in this paper refer to the molecules being manipulated by DE, whether they are proteins, DNA or RNA.

reason for this may be that the methodology used in GA learning is usually quite different from that used in DE. The former usually runs for hundreds or even thousands of generations whereas the latter usually only runs for a few (rarely more than 10) iterations.

Another difference between GAs and DE is that standard GAs usually incorporate relatively weak selection pressure. Common methods for selecting parents are fitness-proportional (de Jong 1975) and tournament selection (Goldberg and Deb 1991). While the severity with which these methods are applied may be varied, both methods typically allow a substantial proportion of offspring to be generated from parents that are somewhat less fit than the 'best' parent. In contrast, DE practitioners usually carry out strict filtering before performing the next mutational step. The filtering step typically allows a small percentage of proteins through to the next stage and in some cases only a single protein is allowed to reproduce. This type of selection is rarely used within GAs although there are a few exceptions (Mühlenbein and Schlierkamp-Voosen 1993), which have generally been influenced by the related field of evolutionary strategies (Bäck *et al.* 1991). One of the questions we hope to answer in this study is whether such high selection pressure is likely to be beneficial within DE.

There are a number of questions that are frequently asked within the GA community that are also asked by DE practitioners. Most of these are part of experimental design within DE but in the context of GAs are described as 'control parameters' or 'hyper-parameters'. Some examples are the choice of

mutation rate, screen size and the use of recombination. The effect of each of these choices is examined in this study. Early work in DE applied low mutation rates, of the order of one mutation per protein, with the assumption that evolution was unlikely to make beneficial mutations at higher mutation rates (Arnold 1996). However, more recently, improved results have been achieved with much higher mutation rates – between 3 and 30 per protein (Zaccolo and Gherardi 1999; Daugherty *et al.* 2000; Reetz 2003; Drummond *et al.* 2005). The choice between high- or low-throughput screening is related to the question of optimal mutation rate: it has been suggested that higher mutation rates are appropriate when a larger screen size is used, but that lower mutation rates are necessary when using a small screen (Arnold 1996; Voigt *et al.* 2001). Recombination has been shown to be useful in a number of studies (Stemmer 1994; Moore *et al.* 1997; Zhang *et al.* 2002; Rowe *et al.* 2003) and examining the value of recombination within a range of algorithms is a further aim of this study.

DE may be contrasted (Arnold 1996) with rational protein design. Ideally the rational approach involves the creation of a realistic model of the mechanism by which a protein interacts with the system under study. In some areas, such as protein folding, substantial progress has been made in understanding mechanisms, although simplifying approximations have to be made in order to produce computationally tractable models (Cowperthwaite and Meyers 2007). Unfortunately, our knowledge of protein folding is not easily translated into predictions of functional performance (Alviso *et al.* 2007) because most

reaction mechanisms are insufficiently well understood and/or too complex to model with reasonable accuracy (Arnold 2001).

Given the limited applicability of rational approaches, methods based upon search algorithms have flourished. These algorithms owe much to machine learning and statistics and fall into two classes. The first is a fully evolutionary approach, which has been described as 'blind' (Arnold 1998) to draw attention to the analogy with the process of natural evolution, popularised as the 'blind watchmaker' (Dawkins 1986). This approach is close to that used in GAs: search is guided by the processes of reproduction, variation and selection only.

The second is a model-based approach, in which a mathematical model of the relationship between protein sequence and function is constructed. This model may then be used to predict promising sequences. As far as the authors are aware, all existing model-based approaches that have been used in DE are stepwise. Each iteration involves the mutation of local sites (single bases or a small number of bases). Once the 'optimum' amino acid or acids have been identified at a particular site, this site is then fixed. A number of approaches fall into this general category, including Reetz's CAST method (Reetz *et al.* 2006), Fox's ProSAR (Fox *et al.* 2003; Fox *et al.* 2007) and Iwakura's Quasi-additive Adaptive Walking (QAW) (Iwakura *et al.* 2006). These methods select bases at each position independently. However, covariation analysis suggests that there are significant correlations between residue frequencies within evolved proteins (Pritchard *et al.* 2001). It is

apparent that stepwise approaches generate different sequences from those generated by evolutionary methods.

Stepwise approaches are usually considered to be a branch of DE. However, they have similarities to the rational design of proteins. Both attempt to identify active sites and hence reaction mechanisms and to use this information to select which proteins to screen. Experiments that use a stepwise algorithm might be better described as ‘semi-rational’ to distinguish them from both fully evolutionary experiments and from fully ‘rational’ approaches.

The task of identifying mechanisms is clearly an important one. However, the stepwise approach has a potential flaw: fixing bases after each step runs the risk of becoming trapped in a local optimum (Arnold 1996), particularly in situations where there are epistatic interactions between amino acids and hence many local optima. A second drawback of semi-rational approaches is that they require the sequencing of proteins, which can be time-consuming and expensive. If semi-rational methods demonstrate superior performance to evolutionary methods the extra financial and time costs may be justified. The final task that we hope to perform through *in silico* simulations is a comparison between the capacities of evolutionary and stepwise methods, to ascertain whether the latter are likely to result in more rapid progress, and if so under what conditions.

To summarise, the aim of this study is to answer the following questions:

- How important is selection pressure in DE?

- What are the optimal mutation rates in DE?
- Does recombination play a useful role in DE?
- Are different approaches appropriate for high-throughput and low-throughput scenarios?
- In what situations is a full evolutionary approach superior to a stepwise approach?

A small number of previous studies have modelled DE or similar methods. Moore and Maranas (Moore and Maranas 2000) constructed quantitative models to predict the probability of producing a specific nucleotide sequence from a number of cycles of error-prone PCR or DNA shuffling. This approach has been generalised to allow for variable amplification efficiency and initial population size (Pritchard *et al.* 2005). Fox demonstrated ProSAR models on NK-landscapes with low K (Fox *et al.* 2003; Fox 2005). Corne *et al* (Corne *et al.* 2002) compared the performances of GAs using a range of mutation rates and of one with a variable mutation rate in the presence of strong selection pressure, as used during DE, using the MAX-ONES function as a test problem. They found that high rates of mutation (between 2/L and 4/L) were beneficial, with a variable rate resulting in further improvements in performance. One aim of this study is to ascertain whether such high mutation rates are also usefully applied to the landscapes commonly encountered during DE.

2 Protein landscapes and NK-landscapes

We believe that knowledge obtained by examining the processes of variation and selection *in silico* using synthetic models can inform *in vitro* experimental

design, but only if the *in silico* simulations have control parameters and a dataset that closely match those of the *in vitro* experiment. Producing a GA with control parameters that closely match the experimental design used in DE is fairly straightforward. Choosing a dataset with properties that mimic those of proteins is more difficult. In particular it is important that the relationships between data are similar to those between proteins. Proteins are known to display epistatic behaviour. In this study we use Kaufmann's NK-landscapes as a model of the protein 'landscape' (Kauffman 1989). In this model, each genotype is a binary string of length N. The fitness of the genotype is calculated as the average fitness of each allele within the gene. However, each allele's fitness is affected by the values of K other alleles. In some NK models epistatically linked alleles must be adjacent. However, we use a model in which the alleles may be anywhere within the bit-string, so incorporating the long-range interactions that have been observed in proteins (Reetz 2004). The value of K may be varied, thereby giving rise to non-epistatic, smooth landscapes or more epistatic, rugged landscapes.

NK-landscapes represent only a subset of all possible epistatically linked landscapes (Heckendorn and Whitley 1997; Heckendorn *et al.* 1998). NK_P-landscapes are a superset of NK-landscapes, in which a proportion, P, of allele combinations make no contribution to the overall fitness (Barnett 1998). By increasing the value of P, the 'neutrality' of landscapes can be increased. However, it has been shown that varying the value of P does not affect the 'juxtapositional complexity' (Smith and Smith 1998), with the result that P will have little effect on the relative performance of different GAs. For this reason,

NK_P landscapes are not considered in this study. For the interested reader, a number of further extensions to the NK model have been suggested (Smith and Smith 2000; Geard *et al.* 2002; Aguirre and Tanaka 2004).

There is a lack of agreement concerning the extent to which epistasis affects the linkage between protein sequences and behaviour. It has been observed that the effects on enzyme activity due to mutations of dihydrofolate reductase were approximately additive (Aita *et al.* 2001; Iwakura *et al.* 2006). Govindarajan *et al* (Govindarajan *et al.* 2003) found that the enzymatic activity of a series of subtilisin variants showed little epistatic interaction. However, the criterion for the identification of epistasis was very strict (p-value < 0.001). Presumably weaker but still significant epistatic interactions occurred between a much larger number of pairs of sites.

Other authors have found that the protein landscape is highly epistatic (Kauffman and Weinberger 1989; Schaeffer *et al.* 2003; Bershtein *et al.* 2006). Further, some protein landscapes are highly anisotropic. Hayashi *et al* (Hayashi *et al.* 2006) evolved a defective phage with the aim of increasing infectivity and found that most of the landscape was smooth and easily searched but that the most important areas, i.e. those containing the most highly infectious phages, were much more rugged. They suggested that different approaches might be required to search the smoother and more rugged parts of the landscape, such as the use of a lower mutation rate for the more rugged sections.

The studies that identified low epistasis considered only naturally occurring mutations whereas those that observed higher epistasis included synthetic variants. The low level of epistasis within naturally occurring proteins has been explained by the observation that they are likely to occur in additive or neutral areas of the protein landscape in which mutations are unlikely to be fatally deleterious (Govindarajan *et al.* 2003). However, DE often aims to produce proteins with properties that have *not* been selected for by natural evolution and should therefore jump out of a local optimum (or neutral network) (Arnold 2001). This will require higher mutation rates which may enable the protein to move away from the local optimum and hence discover improved functional performance (Eigen and Schuster 1978). However, this movement is likely to produce proteins in more epistatic regions of sequence space (Bershtein *et al.* 2006).

Given the lack of certainty concerning the level of epistasis in protein landscapes, we have performed a series of simulations on landscapes with varying levels of epistasis, with values of K varying between 0 (totally smooth) and 10 (very rugged). By using a variety of mutation rates we test whether different mutation rates are optimal for smooth and rugged landscapes.

NK-landscapes have been used previously as a model of protein activity landscapes. Aita and Husimi (Aita and Husimi 1998) used adaptive walks on NK-landscapes to identify the optimum search strategy. At each generation this involved random mutations followed by selection of the best sequence. They found that the use of small screen sizes was effective when searching

smooth landscapes but that it could lead to stagnation within rough landscapes, i.e. entrapment within local optima. Kauffman and Macready (Kauffman and Macready 1995) assessed the efficacy of mutation, recombination and 'pooling' strategies in generating proteins with enhanced fitness using the NK-model. Pooling strategies involve the creation of separate pools of proteins in which each pool has certain amino acids fixed. At each iteration the best pool is selected and sub-pools are created by fixing the amino acids at further positions. According to our definition, pooling is therefore a stepwise algorithm. Kauffman and Macready observed that fixing bases through the selection of a single 'pool' could be detrimental. They showed that the pooling approach could be improved through the introduction of a hill-climbing procedure involving recombination and/or mutation after the selection of each pool or sub-pool. Fox *et al* (Fox *et al.* 2003) reached the opposite conclusion when comparing their ProSAR algorithm with an evolutionary algorithm in their performances when searching NK-landscapes, finding that ProSAR gave better results for K≤3. The contrasting results may be explained by the fact that the two studies used different algorithms. Kauffman *et al* generate libraries using the mutation and recombination operators and select the best variant as the starting point for the next step. Fox *et al* on the other hand do not use the standard mutation and recombination operators but generate libraries using a biased random selection of each allele. Muñoz and Deem (Muñoz and Deem 2008) used an NK-model to demonstrate that it is better to search a large library sparsely than a small library thoroughly, using GAs that ran for 100 generations. This is related to the setting of mutation rates: for a fixed screen size, a high mutation

rate implies searching a large library sparsely while a low mutation rate results in thorough search of a small library.

One aim of this study is to re-evaluate the lessons learnt in previous studies. Like Aita and Husimi, we compare the effects of large and small screen sizes. We extend their study through the introduction of recombination and by allowing more than one protein to contribute to producing the next generation. We attempt to cast light on the debate between evolutionary and stepwise methods by using algorithms that are very similar to the stepwise and evolutionary algorithms used in DE experiments. Finally, we attempt to identify the optimum mutation rate when only a small number of generations are available.

Although NK-landscapes have been widely used as models of protein landscapes, care must be taken when interpreting the results. The 'No Free Lunch Theorem' (NFL) demonstrates that all search algorithms are equally effective when averaged over the space of all possible problems (Wolpert and Macready 1997), at least for single-objective optimisation problems. One consequence of this is that no single algorithm is optimal for all possible problems: an algorithm that has been optimised to solve one problem may be sub-optimal for other problems. The implication of NFL is that optimising an algorithm to search NK-landscapes does not guarantee optimal behaviour on protein landscapes. However, NFL may be partially circumvented by considering landscapes that are similar to each other. There is considerable evidence that NK-landscapes have similar structural features to protein

landscapes and that search algorithms perform similarly when faced with these, respectively, real and synthetic landscapes (Kauffman 1993). We believe that conclusions drawn from this theoretical study may therefore be used to guide the selection of experimental parameters, even though they do not guarantee optimal results.

3 Method

This research is composed of two parts. The main part is the simulation of DE using GAs with a variety of control parameters. These control parameters (screen size, selection pressure, mutation rate, use of crossover) were chosen to cover the range of values used in 'real-world' DE experiments, as described below. All simulations were run on a number of landscapes each with a different ruggedness (K value). The second part of the study involved the simulation of a stepwise algorithm searching on the same landscapes, in order to allow comparisons between the fully evolutionary and semi-rational approaches.

In order to maximise the relevance of our simulations we have chosen ranges of experimental parameters that reflect the values used in 'real' DE experiments. A large number of different techniques are available for generating protein libraries from parent proteins. A recent review (Lutz and Patrick 2004) listed 12 'new technologies for generating molecular diversity', which generate between 100 and 3,000,000 proteins during each generation. Table 1 summarises the parameters used in a number of studies over the last 15 years. It may be seen that screen sizes are typically in the thousands with a few studies using smaller libraries containing hundreds of proteins while a

small number use larger libraries containing hundreds of thousands or even millions of proteins.

When running our GAs we chose two different library sizes that roughly span the range used in DE experiments. For our low-throughput simulation we used a library size of 120. This is close to the smallest libraries that have been used experimentally and is the same size as the library used by Fox when demonstrating ProSAR (Fox *et al.* 2003), thereby making possible direct comparisons with this technique. For the high-throughput simulation we used a library size of 40,000. This value is representative of large screen sizes as indicated by Table 1 and simulations may be run *in silico* within a reasonable timescale with this screen size. The upper limit on library sizes is increasing each year as a result of technological advances. However, parallel increases in computer speeds mean that running larger simulations that reflect this change should become viable in the future.

Proteins were represented in the simulation by binary strings of length N, with N=100 for the high-throughput regime and 40 for the low-throughput regime. For the high-throughput regime, K took values from the set {0, 2, 5, 7, 10}. The low throughput regime was applied to landscapes with K = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}.

From Table 1 it is seen that the number of proteins that are successful in passing through a screen, i.e. proceed to the next generation of reproduction, is generally small and often just one protein is allowed to generate the next

generation. In the evolutionary computing community selection pressures are usually kept relatively low. Low selection pressures are used in order to avoid premature convergence as a result of loss of diversity which can occur on runs of a large number of generations (hundreds or thousands) (Reeves and Rowe 2002). In DE, in contrast, very strong selection pressure is usually applied in order to obtain a significant improvement in performance within a small number of generations. One aim of this study is to assess whether such selection pressure is beneficial. Selection pressure is varied using a (μ,λ) algorithm (Beyer 2001), in which μ and λ are integers, with μ<λ. In the first generation λ individuals are generated and the best μ are chosen to reproduce the next generation. From these μ sequences, parents are selected at random to reproduce λ 'offspring'. The selection and reproduction routines are repeated in subsequent generations. The value of λ is set to a fixed value (40000 for high-throughput and 120 for low-throughput simulations), representing a constant screen size. Selection pressure is varied by changing the value of μ: lower μ indicates stronger selection pressure, i.e. more strict filtering of parents. The range of μ values used is given in Table 2.

The (μ,λ) algorithm is very close to the procedure usually followed during DE. In order to provide a benchmark against which to compare the results with this algorithm we also ran a standard GA using 'tournament selection'. In this algorithm all parents had the opportunity to produce offspring. 4 parents were chosen at random and the parent with the highest fitness was selected to reproduce. During recombination the second parent was chosen in the same way. The resulting offspring passed into the next generation. The procedure

was repeated until the new generation had the same population size as the previous generation. This may be contrasted with the (μ,λ) algorithm in which offspring are over-produced and then culled. Crossover was applied with a probability of 0.6 and mutation at a rate of 1/L, where L is the length of the bit-strings (equal in value to N). These parameters are known to perform well over a range of fitness landscapes (Bäck 1996). The number of proteins in the GA 'population' was generally equal to the screen size used in the (μ,λ) algorithm, i.e. 120 during low-throughput and 40000 during high-throughput simulation. In order to evaluate the improvement in performance due to the (μ,λ) and stepwise algorithms, standard GAs were also run in the low-throughput regime with a range of larger population sizes, up to 4000.

In most evolutionary computing applications, GAs are run to 'convergence', i.e. until no more improvement is observed and most of the population diversity has been lost as a result of repeated selection. However, due to time and money limitations, DE is rarely run for more than 10 generations (see Table 1). In order to model DE experiments realistically our simulations therefore ran for 10 generations.

From Table 1 it may be seen that a wide range of mutation rates have been applied during DE experiments reported in the literature, ranging from less than 1/L up to 27/L. Approximately half of the studies identified used some from of recombination; the others relied entirely on mutation to produce variation. The range of values used within this study have been chosen to reflect the variation in experimental design. All of our experiments were

performed both with and without crossover and with a range of different mutation rates from the set {0.1, 0.2, 0.4, 0.6, 0.8, 1, 2, 3, 4, 5, 10, 15, 20, 25, 30}.

The performance of a stepwise algorithm was compared with the fully evolutionary algorithms. This algorithm was based on ProSAR and uses Partial Least Squares (PLS) models (Wold 1966) to determine which areas of the protein landscape to search. PLS creates a linear mathematical model relating the genotype alleles (0 or 1) to the overall fitness. Coefficients are set by simultaneously regressing onto the dependent and independent variables, thereby avoiding the overfitting which tends to occur when basic regression is applied to a problem with a large number of input variables. ProSAR was primarily designed to operate on small libraries. Furthermore, the computational requirements of PLS make it impractical for large libraries. For these reasons we have only applied it within the low-throughput methodology. In the first generation, sequences were generated randomly. A PLS model was constructed and the 4 positions that had the greatest influence on fitness, i.e. those with the largest PLS coefficients, were set to their optimum values (0 or 1). In subsequent generations only those positions that had not been fixed were allowed to vary. In each generation 4 more positions were fixed, so that at the end of 10 generations all 40 positions had been determined.

For each NK pair 100 landscapes were independently generated. For each algorithm and set of mutation/crossover rates, independent runs were performed on each landscape and the results were averaged across all 100

runs. The best fitnesses achieved and optimal mutation rates were recorded for each algorithm. Overall, over 180,000 runs were performed.

The crossover and mutation operators are described using pseudo-code in the Appendix, as are the standard GA, (μ,λ) and stepwise algorithms used within this study.

4 Results

4.1 High throughput

Figures 1 and 2 show the performance of a standard GA with and without crossover, respectively, for landscapes of different ruggedness. The smoothness of these and later plots reflects the precision attainable in these analyses. Table 3 gives the best fitness achieved on each landscape. It indicates that crossover enhances the performance on smoother landscapes but is detrimental for more rugged landscapes. In order to give an impression of the scale of the improvements, the best fitnesses achieved using random selection of sequences is also included in Table 3.

The introduction of stronger selection pressure via the (μ,λ) algorithm improves performance considerably: the improvement over random selection is approximately doubled when the (μ,λ) algorithm is used for all except the completely smooth landscape (K=0). Figures 3 and 4 show the best fitness achieved on each landscape with a moderate increase in selection pressure (μ = 4000) with and without crossover, respectively.

Table 4 gives the best fitnesses achieved and the corresponding mutation rates. Apart from the overall improvement in fitnesses compared to the standard GA, there are 3 points to note:

- The optimum mutation rates are similar for the (μ,λ) algorithm and the standard GA algorithm.
- The advantage of incorporating crossover is found to apply up to a greater level of ruggedness (K≤5 rather than K≤2) when greater selection pressure is introduced.
- Introduction of crossover reduces the sensitivity to mutation rate. When mutation is the only operator it is very important that the mutation rate is set to its optimum value, but when crossover is present mutation acts as a secondary operator (Maynard Smith 1978; Ochoa *et al.* 1999).

A comparison across different landscapes suggest that the maximum fitness attained is similar for all landscapes with K>0. The *global* optimum for different landscapes has been observed to be independent of K in one study (Smith and Smith 1998). More recent work suggests that the global optimum increases with K, but search algorithms may not reflect this, due to the increased difficulty of searching more rugged landscapes (Skellett *et al.* 2005). The average maximum global fitness for a K=0 landscape is predicted to be 2/3 and it appears that this is attained when crossover is used.

The consequences of further increasing the selection pressure were investigated by decreasing μ. The best fitnesses achieved on the K=5 landscape are given in Table 5. It is clear from these results that

- Performance peaks at a selection pressure of between 1000 and 10000, i.e. μ=40 to 4, with selection pressure defined as λ/μ. Both higher and lower selection pressures give lower fitnesses.
- There is a fairly constant optimum mutation rate of approximately 3/L.
- The introduction of crossover gives a slight edge to the performance of the (μ,λ) algorithm.

Very similar results are achieved by applying strong selection pressure to landscapes with K=0, 2, 7 and 10. For all of these landscapes a selection pressure of 1000 was found to be optimal or near-optimal. Crossover was seen to be beneficial for the smoother landscapes (K≤5) but detrimental for the more rugged landscapes.

The differences between the mean fitness values were shown to be statistically significance using a series of 2-way analysis of variance (ANOVA) tests. For each value of μ and each landscape, p-values were obtained by conditioning on mutation rate and presence/absence of crossover. All p-values were below 0.05 indicating that the difference in mean values is significant at this level.

4.2 Low throughput

Selection pressure was seen to have a similar effect in the low throughput regime. Tables 6 and 7 show the best fitnesses achieved on various NK-landscapes with and without the use of the crossover operator and with varying selection pressure. The following observations may be made:

- As the ruggedness of the landscape increases there seems to be a slight decrease in the optimum selection pressure. However, a selection pressure of about 40, i.e. a (μ,λ) algorithm with μ = 3 and λ=120, performs well across a wide range of landscapes.
- For smoother landscapes (K up to approximately 6), crossover appears to play a useful role. For more rugged landscapes the mutation only algorithm performs better.

Tables 8 and 9 show the optimum mutation rates for each algorithm. It is clear that the optimum mutation rate increases with selection pressure. However, when the optimal selection pressure is present, i.e. (μ,λ) = (3,120), a mutation rate of 2/L gives good results across a range of landscapes both with and without the use of crossover.

Again ANOVA tests indicated that the mutation rates and presence/absence of crossover had a statistically significant effect (p-value < 0.05) on the mean fitness achieved on each landscape.

4.3 Stepwise algorithm

The best fitnesses achieved with a stepwise approach are given in Table 10. Figure 5 shows the variation in best fitness for stepwise and a variety of evolutionary methods. In order to make comparisons across different landscapes, all fitnesses f have been normalised to f_{norm} using equation (1), in which f(GA,1N) and f(GA,100N) are the average fitnesses achieved using standard GAs with screen sizes of, respectively, 1N and 100N.

$$f_{norm} = \frac{f - f(GA,1N)}{f(GA,100N) - f(GA,1N)} \quad (1)$$

This technique has been borrowed from a paper by Fox introducing ProSAR (Fox *et al.* 2003) and facilitates comparison with the results reported in that paper. In figure 5, the reader's attention is drawn to the results using screen sizes of 3N. It is clear that the stepwise method performs better than a standard GA with this size for relatively smooth landscapes (K ≤6). However, for K=7 the performance of the standard GA is similar to that of the stepwise algorithm and for higher K the GA has superior performance.

The (μ,λ) GA, applied with crossover and the optimum selection pressure of (3,120) also gives much better performance than the standard GA. However, this increase in performance is seen to carry over to more rugged landscapes. Even when K=10, this algorithm performs better than a standard GA with 10 times the screen size (30N rather than 3N), whereas the stepwise algorithm performs much worse than a standard GA with a screen size of N.

4.4 Hitting Times

The results have been presented so far as mean fitness levels achieved within a fixed number of generations. This gives an indication of the expected improvement in outcome for a fixed experimental set-up. An alternative approach is to measure the amount of experimental effort required to reach a target fitness value. The results are provided here for a selected problem in order to illustrate the savings possible through good parameter choice. They are presented as 'hitting times', the number of generations required to achieve the target fitness.

A landscape of medium ruggedness (N=40, K=5) was selected and 4 different algorithms were applied: a standard GA, a (μ,λ) GA with modest selection pressure (μ=12), a (μ,λ) GA with strong selection pressure ((μ=3) and a stepwise algorithm. All GAs used crossover. The target fitness was the mean fitness achieved by the standard GA (0.6942). 100 runs were performed using each algorithm.

When using the standard algorithm, 55 runs achieved the target and the average hitting time was 8.13 generations. Weak (μ,λ) selection pressure increased the number of successful runs to 86 and reduced the hitting time to 6.49. Strong selection pressure gave 90 successful runs, with an average hitting time of 5.66. It is clear that the application of appropriate selection pressure can considerably reduce the experimental outlay required to achieve satisfactory results. The stepwise algorithm had less impact on hitting times, yielding 69 successful runs, with an average hitting time of 8.31 generations.

5 Discussion

The primary aim of this study has been to identify effective DE experimental parameters through the use of *in silico* simulations. Our results show that experimental design is of crucial importance, with some methods giving much faster evolution than others. To summarise our findings:

- Strong selection pressure, implemented via filtering of the 'parents' allowed to reproduce, increases the chances of finding beneficial mutants greatly.

- However, it is possible to apply too much selection pressure: many DE experiments use just one parent in each generation, but it appears to be most effective to select 3 parents in the low-throughput regime and approximately 40, (equivalent to 1 in 1000) in the high-throughput regime.
- Moderately high mutation rates (above 1/L) are beneficial in the presence of strong selection pressure. A mutation rate of 2/L performs well across a wide range of landscapes.
- The introduction of recombination makes the setting of an optimal mutation rate less critical. Recombination is beneficial for relatively smooth landscapes but may be detrimental for more rugged landscapes.
- The stepwise algorithm gives a boost in performance similar to that of the (μ,λ) algorithm for smooth landscapes ($K \leq 2$). However, this advantage falls off rapidly as the ruggedness increases, with catastrophic consequences for the most rugged landscapes.
- The pattern of results was very similar for the high-throughput and low-throughput models. In particular, the use of high mutation rates (above 1/L) was found to be beneficial in low-throughput as well as high-throughput mode.
- An illustrative experiment indicates that the experimental settings that produce sequences with the highest fitness are also likely to give the quickest results. This finding suggests that the lessons learnt from this study may be used to achieve a reduction in experimental costs as well as an improvement in screening efficiency.

These findings on NK landscapes could be used to guide the choice of parameters in DE experiments but we have yet to show that, in practice, these predictions are valid. It is our intention to do this, making use of on-chip technology (Knight *et al.* 2008), with which it would be possible to run a variety of GAs, like those used here, and compare their performance directly in raising an aptamer to a target ligand.

As well as providing a guide to the experimental design of evolutionary methods, this study casts light on the debate between stepwise and fully evolutionary methods within DE. The primary distinction between these two methods is in the selection of mutants to be screened. This issue is of the utmost importance in DE. As Reetz has stated:

Efficient directed evolution is not a matter of generating huge libraries that then require considerable efforts in screening for the desired property. The goal is to create a maximum in structural diversity while minimizing the size of the libraries. (Reetz 2004)

The belief that the selection of mutants to be screened is crucial to the success of DE has been a driving force behind the development of model-based methods. However, our investigations suggest that such methods may not be useful and may even be detrimental. Looked at from the opposite point of view they show the power and resilience of evolutionary methods. Of course, the poor performance of stepwise methods compared to evolutionary

methods does not show that all model-based approaches are flawed but the authors believe that it indicates a significant problem with certain types of modelling. There are two likely sources of the problems observed with the stepwise approach used here. The first is that it fixed amino acids at each iteration, without allowing backtracking. The absence of backtracking is likely to cause particular problems in the presence of epistasis. Epistasis results in local optima and a procedure that does not include backtracking is likely to become trapped on one of these optima (Liebeton *et al.* 2000; Reetz *et al*. 2006). The second likely source of inaccuracies is the use of a linear model to approximate non-linear interactions. Again this will cause problems when the contributions of individual amino acids are non-additive.

The presence of appropriate selection pressure has been shown to greatly enhance the evolutionary process. Stepwise methods also have strong selection pressure, since library mutants are guided towards the regions of the protein space which the model indicates are best. Evolutionary methods concentrate on promising areas of protein space in a similar way. However, the selection of mutants occurs via the selection of parent proteins rather than through the accumulation of ‘knowledge’. It seems likely that the presence of strong selection pressure is essential during DE, given the small number of iterations that are generally available (Bäck 1996). Whether evolutionary or model-based methods are superior depends upon whether this selection pressure is selecting the best areas on which to concentrate the search procedure. Our results show that fully evolutionary methods with strong selection pressure are robust, giving good performance across a range of

problem landscapes. The model-based approach used here appears to be more fragile, performing well for smooth landscapes but less well for medium and rugged landscapes.

While we have shown that model-based approaches should be used with care within the evolutionary process itself, we believe that they have two useful roles. The first is in choosing sites to be targeted for mutation. The incorporation of computational methods into this choice has been shown to be beneficial (Wong *et al.* 2007). However, the implication of our results is that simultaneous mutations at multiple sites are likely to be more successful than stepwise mutations at individual sites. The second use of models is in trying to describe a protein landscape *after* it has been explored using evolutionary methods.

6 Conclusions and Future Work

We have shown that simulating the Directed Evolution process *in silico* can give clear answers to questions concerning preferred experimental design. Based on our simulations, the following tentative recommendations may be made:

- Strong selection pressure is highly advantageous during Directed Evolution. However it is possible to have too much selection pressure: selecting a single parent at each generation is likely to be detrimental.
- A high mutation rate – 2/L to 3/L – combines well with strong selection pressure.

- Recombination is valuable in the presence of low or medium epistasis but detrimental in highly epistatic protein landscapes (K≥5).

These conclusions seem to be quite general. The advantages of strong selection pressure and fairly high mutation rates have been shown to operate on landscapes with highly-varied levels of ruggedness, operating in either high-throughput or low-throughput mode.

A model-based approach has been seen to perform poorly compared to an evolutionary approach on landscapes of medium and high ruggedness. It is likely that a different type of modelling, particularly one incorporating non-linearities, could well perform better. However, fitting the type of model to the observed protein landscape may be non-trivial. The two main issues to be considered are the selection of the model to be used and the way in which selection of library mutants is performed. The model used here is linear in nature and is therefore likely to perform well only when the contributions of individual amino acids are roughly additive. The problem may have been exacerbated by the selection procedure, which fixed a number of bases at each iteration. We would not recommend the fixing of individual amino acids based on model predictions, since this is likely to lead to stagnation at a local optimum.

While care needs to be taken when using a model to select a mutant library, there are two situations in which modelling is undoubtedly useful. The first is that where the mechanism of the target protein function is well understood. In this situation a model could introduce constraints that restrict the search

space to those proteins that are known to fulfil the desired function well. A strategy such as this could combine 'rational' design with evolutionary methods (Wong *et al.* 2007; Xiong *et al.* 2007). The second role for modelling is in improving our understanding of mechanism *after* performing Directed Evolution. At the end of a DE run, a large number of proteins will have been sequenced and an assessment made of their ability to perform some desired function. From this information it is possible to construct models that indicate reactive sites and epistatic links. This information could support particular mechanisms of action.

In future work we intend to investigate the extent to which predictions for optimal experimental design settings derived from *in silico* investigations may be applied to *in vitro* experiments. We also intend to explore further the uses (and limitations) of mathematical models within two areas of the Directed Evolution process: library generation and post-evolution explanation. The focus of our attention is likely to be on non-linear modelling techniques such as Random Forests (Breiman 2001) and Genetic Programming (Koza 1992; Kell 2002).

Acknowledgments

David Wedge is supported by a grant from the UK Home Office and William Rowe by a grant from the BBSRC. Joshua Knowles is supported by a David Phillips Fellowship from the BBSRC.

Appendix: Algorithms used

The mutation operator was applied to each protein using the following procedure:

1. Select a protein using tournament selection.
2. Set current bit = 1.
3. Generate a random number between 0.0 and 1.0.
4. If the random number is greater than the mutation rate, ‘flip’ the current bit.
5. Set current bit = current bit + 1.
6. If current bit > bit length, exit, else go to 3.

Uniform crossover was used. This operator used the following procedure:

1. Select 2 parents using tournament selection.
2. Set current bit = 1.
3. Generate a random number between 0.0 and 1.0.
4. If the random number is greater than 0.5 assign the offspring the bit at the current bit position from parent 1, else assign the corresponding bit from parent 2.
5. Set current bit = current bit + 1.
6. If current bit > bit length, exit, else go to 3.

The standard GA had the following control parameters:

Mutation rate =1/L

Crossover rate = 0.6

Selection method = 4-fold tournament

It used the following procedure:

1. Generate a population of p sequences randomly.
2. Set generation_no = 1.

3. Set current_sequence = 1 and new_population to null.
4. Select parent(s) from the population using 4-fold tournament selection
5. Generate offspring from parents using crossover/mutation operators
6. Add offspring to new_population.
7. Set current_sequence = current_sequence+1
8. If current_sequence ≤ p go to 4.
9. Set population = new_population
10. generation_no = generation_no +1.
11. If generation_no > 10, exit, else, go to 3.

The (μ,λ) algorithm used the following procedure:

1. Generate a population of λ sequences randomly.
2. Set generation_no = 1.
3. Select the best μ sequences from the current population as parents.
4. Set current_sequence = 1 and new_population to null.
5. Select parent(s) randomly.
6. Generate offspring from parents using crossover/mutation operators
7. Add offspring to new_population.
8. Set current_sequence = current_sequence+1
9. If current_sequence ≤ λ go to 5.
10. Set population = new_population
11. generation_no = generation_no +1.
12. If generation_no > 10, exit, else, go to 3.

The stepwise algorithm used the following procedure

1. Generate a population of λ sequences randomly.
2. Create a PLS model.
3. Fix the 4 positions that have the highest PLS coefficients.
4. If all positions are fixed, exit.
5. Vary the unfixed positions randomly.
6. Go to 2.

References

Aguirre, H. E. and K. Tanaka (2004). Insights on Properties of Multiobjective MNK-Landscapes. Proceedings of the Congress on Evolutionary Computation, Portland, OR, USA, IEEE, 196-203.

Aita, T. and Y. Husimi (1998). "Adaptive Walks by the Fittest among Finite Random Mutants on a Mt. Fuji-type Fitness Landscape." Journal of Theoretical Biology **193**: 383-405.

Aita, T., M. Iwakura, et al. (2001). "A cross-section of the fitness landscape of dihydrofolate reductase." Protein Engineering **14**(9): 633-638.

Alviso, O., B. D. Allen, et al. (2007). "Computational protein design promises to revolutionize protein engineering." Biotechniques **42**(1): 31-39.

Arnold, F. H. (1996). "Directed Evolution: Creating Biocatalysts for the Future." Chemical Engineering Science **51**(23): 5091-5102.

Arnold, F. H. (1998). "When blind is better: Protein design by evolution." Nature Biotechnology **16**: 617-618.

Arnold, F. H. (2001). "Combinatorial and computational challenges of biocatalyst design." Nature **409**: 253-257.

Bäck, T. (1996). Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms, Oxford University Press.

Bäck, T., F. Hoffmeister, et al. (1991). A Survey of Evolution Strategies. Proceedings of the International Conference on Genetic Algorithms, San Diego, CA, USA, Morgan Kaufmann, 2-9.

Barnett, L. (1998). Ruggedness and Neutrality - The NKp family of Fitness Landscapes. Proceedings of the sixth international conference on Artificial life, Madison, Wisconsin, USA, MIT Press, 18-27.

Bershtein, S., M. Segal, et al. (2006). "Robustness-epistasis link shapes the fitness landscape of a randomly drifting protein." Nature **444**: 929-932.

Beyer, H.-G. (2001). The Theory of Evolution Strategies. New York, NY, USA, Springer.

Breiman, L. (2001). "Random Forests." Machine Learning **45**(1): 5-32.

Chautard, H., E. Blas-Galindo, et al. (2007). "An activity-independent selection system of thermostable protein variants." Nature Methods **4**(11): 919-921.

Cherry, J. R. and A. L. Fidantsef (2003). "Directed evolution of industrial enzymes: an update." Current Opinion in Biotechnology **14**: 438-443.

Coco, W., M., W. E. Levinson, et al. (2001). "DNA shuffling method for generating highly recombined genes and evolved enzymes." Nature Biotechnology **19**: 354-359.

Corne, D., M. J. Oates, et al. (2002). On Fitness Distributions and Expected Fitness Gain of Mutation Rates in Parallel Evolutionary Algorithms. Parallel Problem-Solving from Nature VII, Granada, Spain, 132-141.

Cowperthwaite, M. C. and L. A. Meyers (2007). "How Mutational Networks Shape Evolution: Lessons from RNA Models." Annual Review of Ecology, Evolution and Systematics **38**: 203-230.

Daugherty, P. S., G. Chen, et al. (2000). "Quantitative analysis of the effect of the mutation frequency on the affinity maturation of single chain Fv antibodies." Proceedings of the National Academy of Sciences **97**(5): 2029-2034.

Dawkins, R. (1986). The Blind Watchmaker. New York, NY, USA, Norton.

de Jong, K. A. (1975). An Analysis of the Behavior of a Class of Genetic Adaptive Systems, University of Michigan. **PhD**.

Drummond, D. A., B. L. Iverson, et al. (2005). "Why High-error-rate Random Mutagenesis Libraries are Enriched in Functional and Improved Proteins." Journal of Molecular Biology **350**: 806-816.

Eigen, M. and P. Schuster (1978). "The Hypercycle: A Principle of Natural Self-Organization." Naturwissenschaften **65**(7): 341-369.

Fox, R. (2005). "Directed molecular evolution by machine learning and the influence of nonlinear interactions." Journal of Theoretical Biology **234**: 187-199.

Fox, R. J., S. C. Davis, et al. (2007). "Improving catalytic function by ProSAR driven enzyme evolution." Nature Biotechnology **25**(3): 338-344.

Fox, R. J., A. Roy, et al. (2003). "Optimizing the search algorithm for protein engineering by directed evolution." Protein Engineering **16**(8): 589-597.

Geard, N., J. Wiles, et al. (2002). A Comparison of Neutral Landscapes - NK, NKp and NKq. Proceedings of the Congress on Evolutionary Computation, Honolulu, HI, USA, IEEE, 205-210.

Goldberg, D. E. and K. Deb (1991). A Comparative Analysis of Selection Scemes Used in Genetic Algorithms. Foundations of Genetic Algorithms 1. G. J. E. Rawlins. San Mateo, CA, USA, Morgan Kaufmann**:** 69-93.

Govindarajan, S., J. E. Ness, et al. (2003). "Systematic Variation of Amino Acid Substitutions for Stringent Assessment of Pairwise Covariation." Journal of Molecular Biology **328**: 1061-1069.

Hayashi, Y., T. Aita, et al. (2006). "Experimental Rugged Fitness Landscape in Protein Sequence Space." PLoS One **1**(1): e96.

Heckendorn, R. B., S. Rana, et al. (1998). Test Function Generators as Embedded Landscapes. Proceedings of the Fifth Workshop on Foundations of Genetic Algorithms (FOGA), Madison, WI, USA, Morgan Kaufmann, 183-198.

Heckendorn, R. B. and D. Whitley (1997). A Walsh Analysis of NK-Landscapes. Proceedings of the seventh International Conference on Genetic Algorithms, East Lansing, MI, USA, Morgan Kaufmann.

Iwakura, M., K. Maki, et al. (2006). "Evolutional Design of a Hyperactive Cysteine- and Methionine-free Mutant of Escherichia coli Dihydrofolate Reductase." Journal of Biological Chemistry **281**(19): 13234-13246.

Joyce, G. F. (1994). "In vitro evolution of nucleic acids." Current Opinion in Structural Biology **4**: 331-336.

Kauffman, S. A. (1989). Adaptation on Rugged Fitness Landscapes. Lectures in the Sciences of Complexity. D. L. Stein, Addison-Wesley. **1:** 527-618.

Kauffman, S. A. (1993). The Origins of Order. Oxford, Oxford University Press.

Kauffman, S. A. and W. G. Macready (1995). "Search Strategies for Applied Molecular Evolution." Journal of Theoretical Biology **173**: 427-440.

Kauffman, S. A. and E. D. Weinberger (1989). "The NK model of rugged fitness landscapes and its application to maturation of the immune response." Journal of Theoretical Biology **141**: 211-245.

Kell, D. (2002). "Genotype-phenotype mapping: genes as computer programs." Trend in Genetics **18**(11): 555-559.

Knight, C. G., M. Platt, et al. (2008). "Array-based evolution of DNA aptamers allows modelling of an explicit sequence-fitness landscape." Nucleic Acids Research: In Press.

Koza, J. R. (1992). Genetic Programming : on the Programming of Computers by means of Natural Selection. Cambridge, Mass. ; London, MIT.

Leung, D. W., E. Chen, et al. (1989). "A method for random mutagenesis of a defined DNA segment using a modified polymerase chain reaction." Technique **1**: 11-15.

Liebeton, K., A. Zonta, et al. (2000). "Directed evolution of an enantioselective lipase." Chemistry and Biology **7**: 709-718.

Lutz, S. and W. M. Patrick (2004). "Novel methods for directed evolution of enzymes:quality, not quantity." Current Opinion in Biotechnology **15**: 291-297.

Maynard Smith, J. (1978). The Evolution of Sex. Cambridge, UK, Cambridge University Press.

Moore, G. L. and C. D. Maranas (2000). "Modeling DNA Mutation and Recombination for Directed Evolution Experiments." Journal of Theoretical Biology **205**: 483-503.

Moore, J. C., H.-M. Jin, et al. (1997). "Strategies for the in vitro Evolution of Protein Function: Enzyme Evolution by Random Recombination of Improved Sequences." Journal of Molecular Biology **272**: 336-347.

Mühlenbein, H. and D. Schlierkamp-Voosen (1993). "Predictive Models for the Breeder Genetic Algorithm, 1. Continuous Parameter Optimization." Evolutionary Computation **1**(1): 25-49.

Muñoz, E. and M. W. Deem (2008). "Amino acid alphabet size in protein evolution experiments: better to search a small library thoroughly or a large library sparsely?" Protein Engineering, Design and Selection **21**(5): 311-317.

Ochoa, G., I. Harvey, et al. (1999). On Recombination and Optimal Mutation Rates Proceedings of the 8th Genetic and Evolutionary Computation Conference (GECCO), Orlando, Florida, USA, Morgan Kaufmann, 488-495.

Piatesi, A., S. W. Howland, et al. (2006). "Directed evolution for improved secretion of cancer-testis antigen NY-ESO-1 from yeast." Protein Expression and Purification **48**: 232-242.

Pritchard, L., P. Bladon, et al. (2001). "Evaluation of a novel method for the identification of coevolving protein residues." Protein Engineering **14**(8): 549-555.

Pritchard, L., D. Corne, et al. (2005). "A general model of error-prone PCR." Journal of Theoretical Biology **234**: 497-509.

Reetz, M. T. (2003). "Controlling the enantioselectivity of enzymes by directed evolution: Practical and theoretical ramifications." Proceedings of the National Academy of Sciences **101**(16): 5716-5722.

Reetz, M. T. (2004). "Controlling the enantioselectivity of enzymes by directed evolution: Practical and theoretical ramifications." Proceedings of the National Academy of Sciences **101**(16): 5716-5722.

Reetz, M. T., L.-W. Wang, et al. (2006). "Directed Evolution of Enantioselective Enzymes: Iterative Cycles of CASTing for Probing Protein-Sequence Space." Angewandte Chemie **118**: 1258-1263.

Reeves, C. R. and J. E. Rowe (2002). Genetic Algorithms - Principles and Perspectives: A Guide to GA Theory. Dordrecht, Kluwer Academic Publishers.

Rowe, L. A., M. L. Geddie, et al. (2003). "A Comparison of Directed Evolution Approaches Using the β-Glucuronidase Model System." Journal of Molecular Biology **332**: 851-860.

Schaeffer, S. W., M. P. Goetting-Minesky, et al. (2003). "Evolutionary genomics of inversions in Drosophila pseudoobscura: Evidence for epistasis." Proceedings of the National Academy of Sciences **100**(14): 8319-8324.

Skellett, B., B. Cairns, et al. (2005). Maximally Rugged NK Landscapes Contain the Highest Peaks. Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), Washington, DC, USA, ACM, 579-584.

Smith, R. E. and J. E. Smith (1998). An Examination of Tunable, Random Search Landscapes. Proceedings of the Fifth Workshop on Foundations of Genetic Algorithms, Madison,WI,USA, Morgan Kaufmann, 165-182.

Smith, R. E. and J. E. Smith (2000). New Methods for Tunable, Random Landscapes. Proceedings of the Sixth Workshop on Foundations of Genetic Algorithms, Charlottesville, VA, USA, Morgan Kaufmann, 47-67.

Stemmer, W. P. C. (1994). "DNA shuffling by random fragmentation and reassembly: In vitro recombination for molecular evolution." Proceedings of the National Academy of Sciences **91**: 10747-10751.

Stemmer, W. P. C. (1994). "Rapid evolution of a protein in vitro by DNA shuffling." Nature **370**: 389-391.

Sylvestre, J., H. Chautard, et al. (2006). "Directed Evolution of Biocatalysts." Organic Process Research and Development **10**: 562-571.

Voigt, C. A., S. L. Mayo, et al. (2001). "Computationally Focusing the Directed Evolution of Proteins." Journal of Cellular Biochemistry Supplement **37**: 58-63.

Wold, H. (1966). Estimation of principal components and related models by iterative least squares. Multivariate Analysis. P. R. Krishnaiah. New York, Academic Press.

Wolpert, D. H. and W. G. Macready (1997). "No Free Lunch Theorems for Optimization." IEEE Transactions on Evolutionary Computation **1**(1): 67-82.

Wong, T. S., D. Roccatano, et al. (2007). "Steering directed protein engineering: strategies to manage combinatorial complexity of mutant libraries." Environmental Microbiology **9**(11): 2645-2659.

Xiong, A.-S., R.-H. Peng, et al. (2007). "A semi-rational design strategy of directed evolution combined with chemical synthesis of DNA sequences." Biological Chemistry **388**(12): 1291-1300.

Yun, C., H. Matsuda, et al. (2006). "Directed Evolution to Enhance Secretion Efficiency and Thermostability of Chitosanase from Mitsuaria chitosanitabida 3001." Bioscience Biotechnology and Biochemistry **70**(2): 559-563.

Zaccolo, M. and E. Gherardi (1999). "The Effect of High-frequency Random Mutagenesis on in Vitro Protein Evolution: A Study on TEM-1 β-Lactamase." Journal of Molecular Biology **285**: 775-783.

Zhang, Y.-X., K. Perry, et al. (2002). "Genome shuffling leads to rapid phenotypic improvement in bacteria." Nature **415**: 644-646.

Table legends

Table 1. *Experimental parameters used during DE*

Table 2. *Algorithms applied to the NK-landscapes*

Table 3. *Best fitnesses and best mutation rates achieved with a standard GA (high throughput). Bold entries indicate whether the mutation-only or with-crossover algorithm is superior. GAs have a population of 40,000 and are run for 10 generations. 'Random' is the best fitness achieved from 400,000 randomly generated sequences.*

Table 4. *Best fitnesses and best mutation rates achieved with a (μ,λ) GA, with μ=4000 and λ=40000. Bold entries indicate whether the mutation-only or with-crossover algorithm gives the higher mean fitness.*

Table 5. *Best fitnesses and best mutation rates achieved with (μ,λ) GAs with varying selection pressure (high throughput, K=5). Bold entries indicate whether the mutation-only or with-crossover algorithm gives the higher mean fitness.*

Table 6. *Best fitnesses achieved with standard and (μ,λ) GAs with crossover (low throughput). Shading indicates the optimum selection pressure for each landscape. Bold entries indicate that crossover is advantageous.*

Table 7. *Best fitnesses achieved with standard and (μ,λ) GAs without crossover (low throughput) Shading indicates the optimum selection pressure for each landscape. Bold entries indicate that crossover is disadvantageous.*

Table 8. *Best mutation rates for standard and (μ,λ) GAs with crossover (low throughput)*

Table 9. *Best mutation rates for standard and (μ,λ) GAs without crossover (low throughput)*

Table 10. *Best fitnesses achieved with the stepwise algorithm (low throughput)*

Figure legends

Figure 1. *Best fitnesses achieved by a standard GA with crossover*

Figure 2. *Best fitnesses achieved by a standard GA without crossover*

Figure 3. *Best fitnesses achieved by a (μ,λ) GA with crossover, with μ=4000 and λ=40000*

Figure 4. *Best fitnesses achieved by a (μ,λ) GA without crossover, with μ=4000 and λ=40000*

Figure 5. *Normalised best fitnesses achieved by various GA algorithms and a stepwise algorithm (low throughput). All included GAs used crossover and the (μ,λ) GA had μ=3 and λ=120. Screen sizes are indicated in brackets, as a multiple of N, where N=40.*

Table 1:

reference	screen size	number selected	number of generations	mutation rate	recombination used?
(Chen and Arnold 1993)	~1300	1	3	unknown	No
(You and Arnold 1994)	~1000	1	7	unknown	No
(Moore and Arnold 1996)	1000-7500	1	4	1.5-4.5	No
(Moore *et al.* 1997)	~1000	~5	6	0.6	Yes
(Reetz *et al.* 1997)	~2000	1	4	1-2	No
(Zhang *et al.* 1997)	10000	20-40	7	~1	Yes
(Yano *et al.* 1998)	4000000	100	5	15	No
(Oue *et al.* 1999)	~50000000	~100	8		No
(Zaccolo and Gherardi 1999)	~40000	1	3	8.2-27.2	No
(Zhao and Arnold 1999)	~5000	1-5	5	2-3	Yes
(Daugherty *et al.* 2000)	~1000000	~10000	4-5	1.7-22.5	No
(Liebeton *et al.* 2000)	800	1	5	~1	No
(Reetz 2000)	2000	1	5	1, 2	No
(Yano and Kagamiyama 2001)	2000000-6000000	~200	50	~4	Yes
(Kaper *et al.* 2002)	2048		3	unknown	Yes
(Oh *et al.* 2002)	~10000	10	2	unknown	Yes
(Bessler *et al.* 2003)	7200	16	2	1	Yes
(Bulter *et al.* 2003)	~2000	1-12	10		Yes
(Ikebukuro *et al.* 2005)	10	5	7	1	Yes
(Johannes *et al.* 2005)	~8000	1	3	1-2	No
(Hayashi *et al.* 2006)	10 - 1000000	1	20	2.4	No
(Piatesi *et al.* 2006)	10000000-30000000	unknown	4	<2, 2-4, >4	No
(Reetz *et al.* 2006)	~3000	1	6	2-3	No
(Fox *et al.* 2007)	~14000	5-10	18	~1	Yes

Table 2:

high throughput	low throughput
standard GA (population 40000)	standard GA (population 120)
GA with (μ,λ) = (4000,40000)	GA with (μ,λ) = (12,120)
GA with (μ,λ) = (400,40000)	GA with (μ,λ) = (6,120)
GA with (μ,λ) = (40,40000)	GA with (μ,λ) = (3,120)
GA with (μ,λ) = (4,40000)	GA with (μ,λ) = (1,120)
GA with (μ,λ) = (2,40000)	
GA with (μ,λ) = (1,40000)	stepwise (population 120)

Table 3:

K	random	mutation only		with crossover	
		best fitness	best mutation rate	best fitness	best mutation rate
0	0.5804	0.6234	4/L	**0.6547**	0.2/L
2	0.6034	0.6655	3/L	**0.6865**	0.1/L
5	0.6229	**0.6785**	2/L	0.6715	0.4/L
7	0.6246	**0.6795**	1/L	0.6681	0.8/L
10	0.6290	**0.6798**	1/L	0.6663	0.8/L

Table 4:

K	mutation only		with crossover	
	best fitness	best mutation rate	best fitness	best mutation rate
0	0.6394	4/L	**0.6670**	0.1/L
2	0.6904	3/L	**0.7188**	0.1/L
5	0.7063	2/L	**0.7080**	0.8/L
7	**0.7072**	2/L	0.7053	1/L
10	**0.7050**	1/L	0.7006	0.8/L

Table 5:

(μ,λ)	mutation only		with crossover	
	best fitness	best mutation rate	best fitness	best mutation rate
(4000,40000)	0.7063	2/L	**0.7080**	0.8/L
(400,40000)	0.7464	3/L	**0.7478**	2/L
(40,40000)	0.7490	4/L	**0.7495**	4/L
(4,40000)	**0.7494**	3/L	0.7488	3/L
(2,40000)	0.7426	4/L	**0.7434**	3/L
(1,40000)	**0.7457**	4/L		

Table 6:

K	standard GA	(12,120)	(6,120)	(3,120)	(2,120)
0	0.6482	0.6680	0.6686	**0.6686**	0.6671
1	0.6744	0.7040	0.7052	**0.7053**	0.6996
2	0.6875	0.7270	0.7286	**0.7308**	0.7178
3	0.6928	0.7329	**0.7376**	0.7355	0.7287
4	0.6981	0.7341	**0.7397**	0.7374	0.7303
5	0.6984	0.7370	0.7350	**0.7382**	0.7313
6	0.6962	0.7305	**0.7337**	0.7323	0.7231
7	0.6935	0.7248	0.7259	0.7269	0.7189
8	0.6944	0.7172	0.7201	0.7225	0.7148
9	0.6904	0.7170	0.7156	0.7178	0.7092
10	0.6874	0.7112	0.7136	0.7135	0.7071

Table 7:

K	standard GA	(12,120)	(6,120)	(3,120)	(2,120)	(1,120)
0	0.6392	0.6613	0.6657	0.6672	0.6659	0.7000
1	0.6659	0.6972	0.7019	0.7038	0.6978	0.6987
2	0.6791	0.7174	0.7240	0.7266	0.7177	0.7162
3	0.6853	0.7272	0.7351	0.7364	0.7256	0.7227
4	0.6901	0.7319	0.7374	0.73492	0.7296	0.7264
5	0.6911	0.7299	0.7355	0.7337	0.7267	0.7260
6	0.6968	0.7263	0.7280	0.7300	0.7232	0.7231
7	0.6902	0.7218	**0.7272**	0.7252	0.7180	0.7157
8	0.6872	0.7187	0.7200	**0.7227**	0.7150	0.7169
9	0.6890	0.7147	**0.7179**	0.7170	0.7119	0.7094
10	0.6862	0.7089	0.7139	**0.7152**	0.7085	0.7039

Table 8:

K	standard GA	(12,120)	(6,120)	(3,120)	(2,120)
0	2	0.6	1	1	2
1	2	0.8	2	3	2
2	1	1	2	1	2
3	0.8	2	2	1	2
4	0.8	0.8	2	2	3
5	0.6	1	2	2	3
6	0.8	1	2	2	2
7	0.8	2	2	2	2
8	0.6	2	2	2	3
9	0.6	1	1	3	3
10	0.6	1	2	1	2

Table 9:

K	standard GA	(12,120)	(6,120)	(3,120)	(2,120)	(1,120)
0	2	2	2	2	2	2
1	2	2	2	2	3	3
2	2	2	2	3	3	3
3	2	2	2	2	2	2
4	2	2	2	2	3	3
5	1	2	2	1	2	3
6	1	2	2	2	2	3
7	1	1	2	2	2	3
8	1	2	2	3	3	2
9	1	2	2	2	3	3
10	1	0.8	2	2	2	3

Table 10:

K	best fitness
0	0.6688
1	0.706776
2	0.727978
3	0.72855
4	0.723386
5	0.716273
6	0.708067
7	0.694728
8	0.685902
9	0.669519
10	0.666328

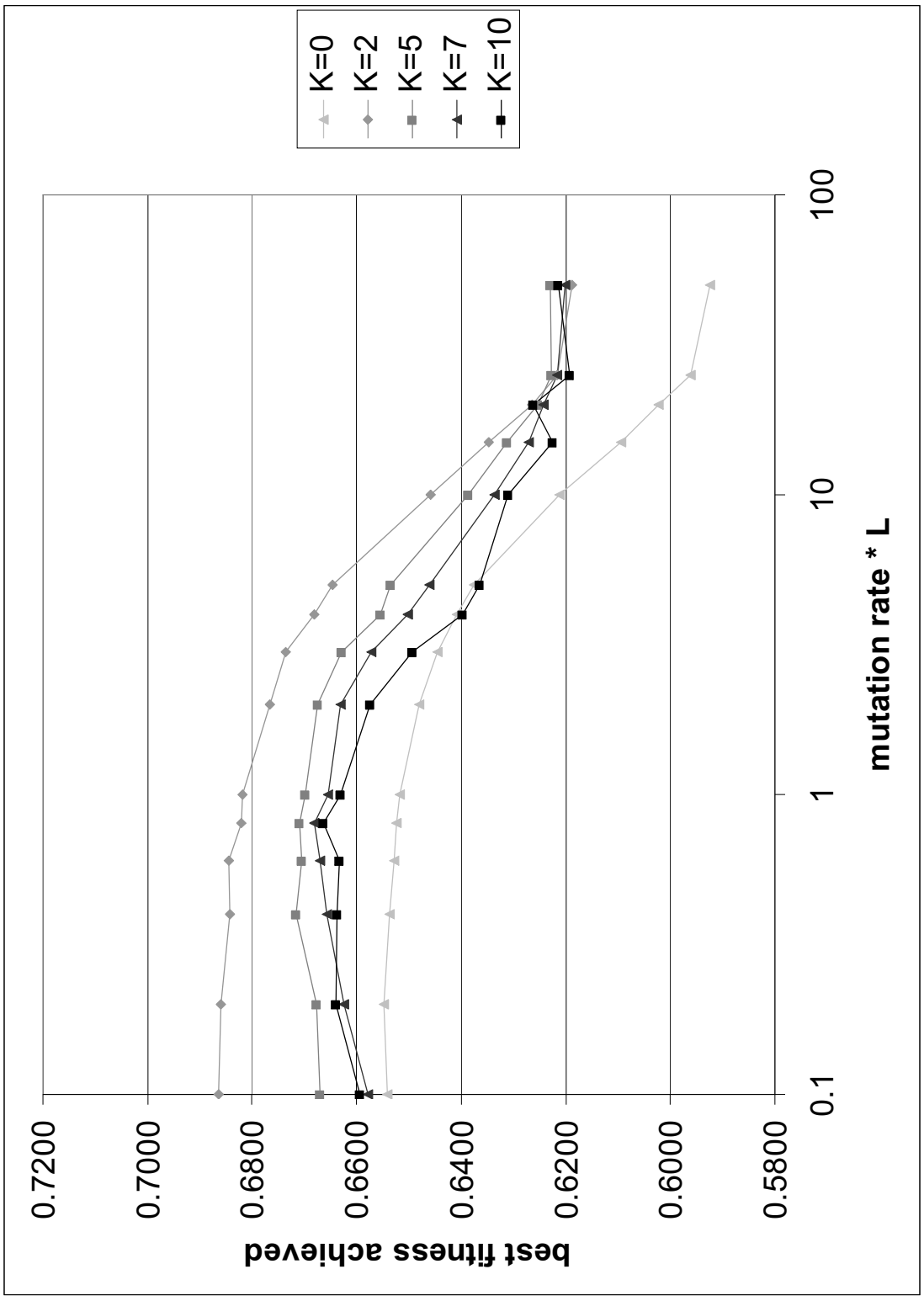

Figure 1

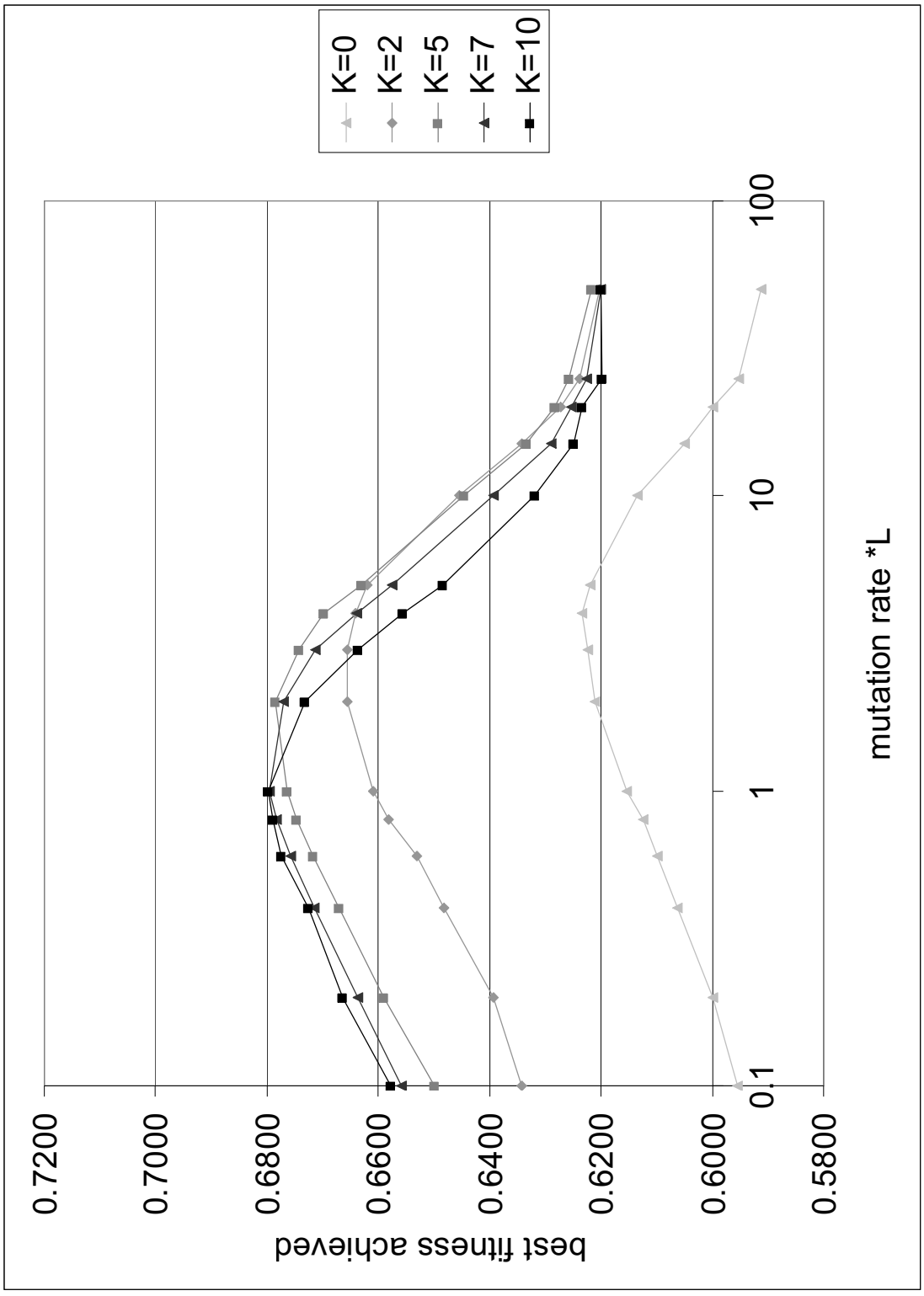

Figure 2

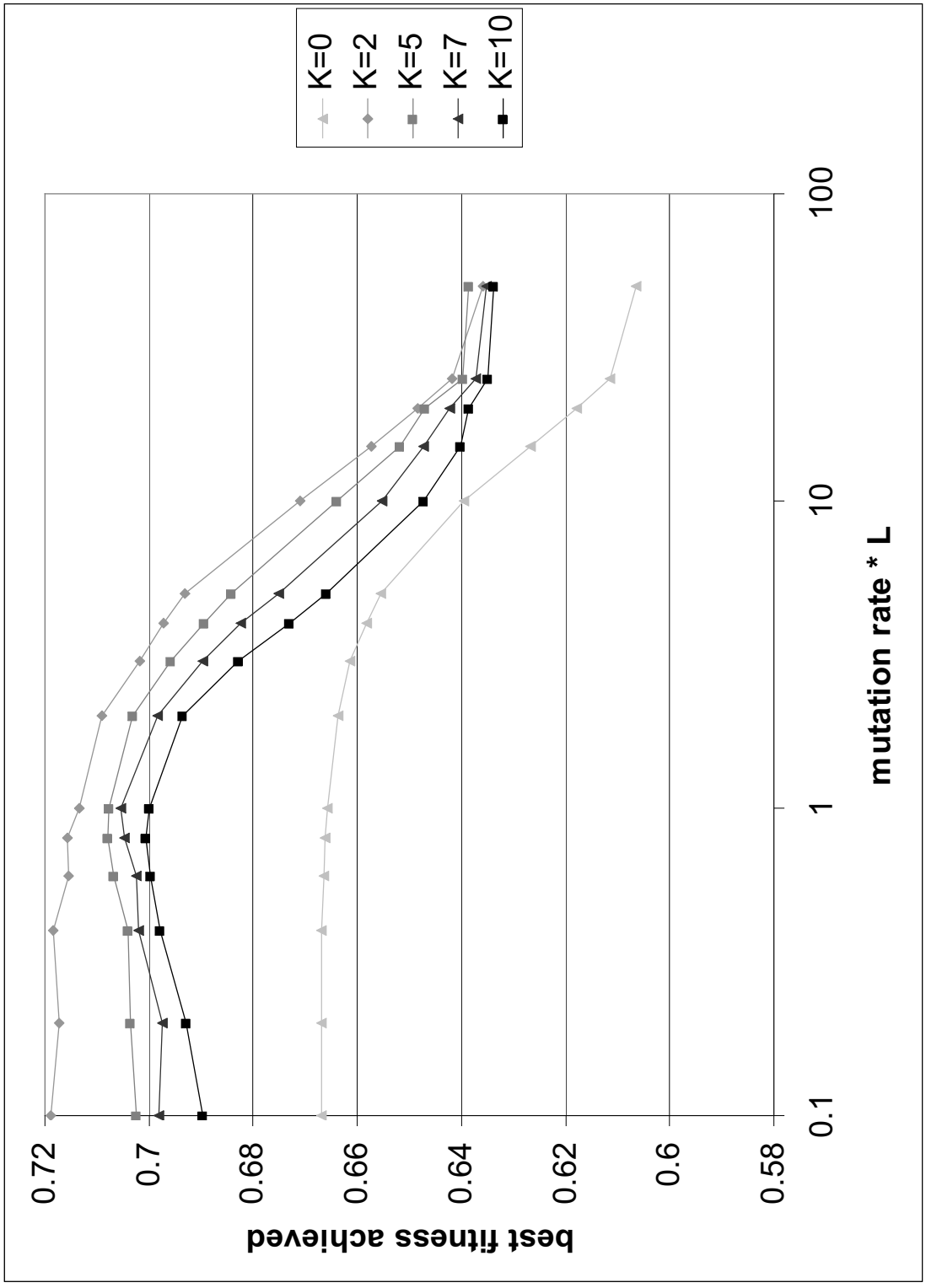

Figure 3

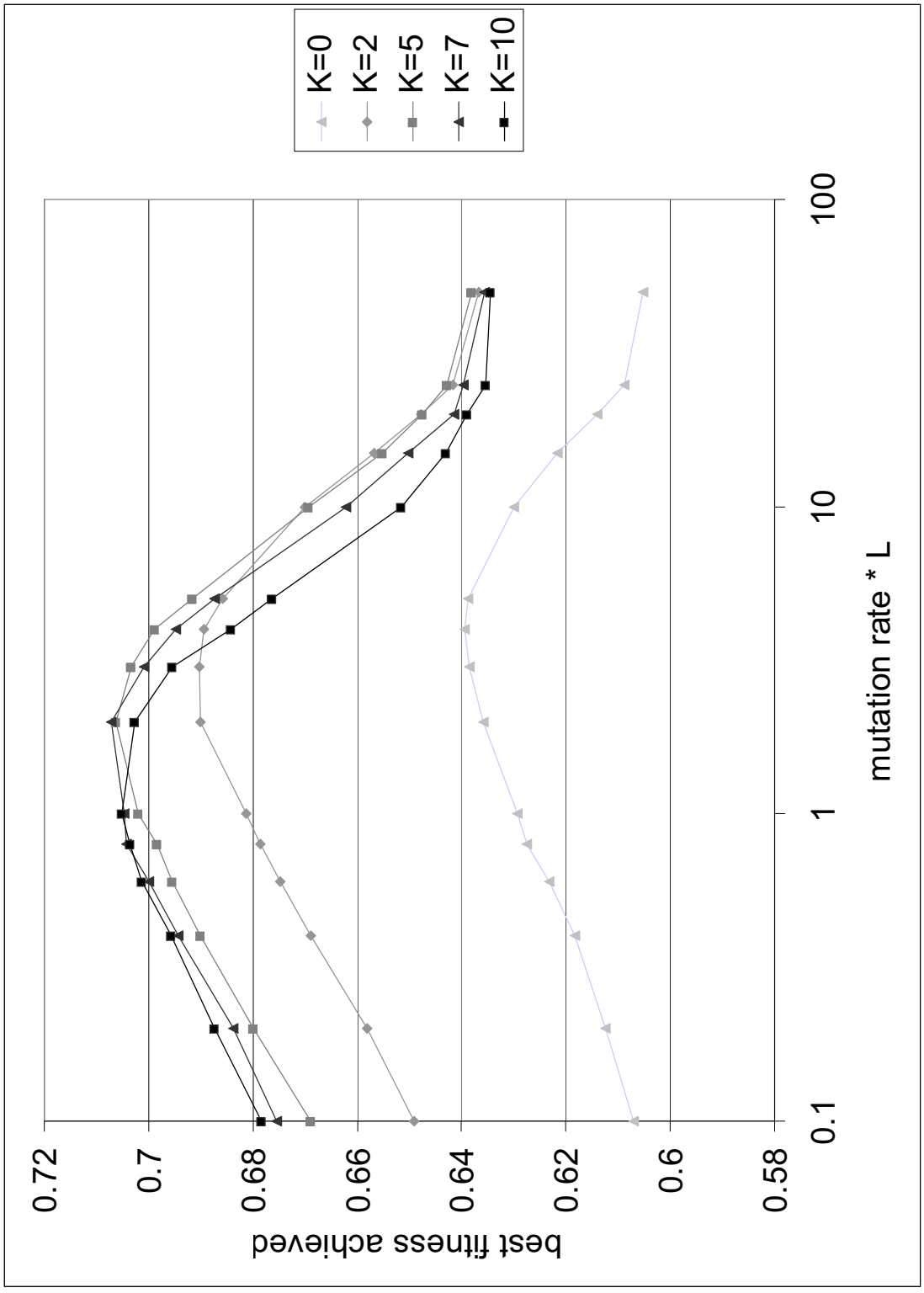

Figure 4

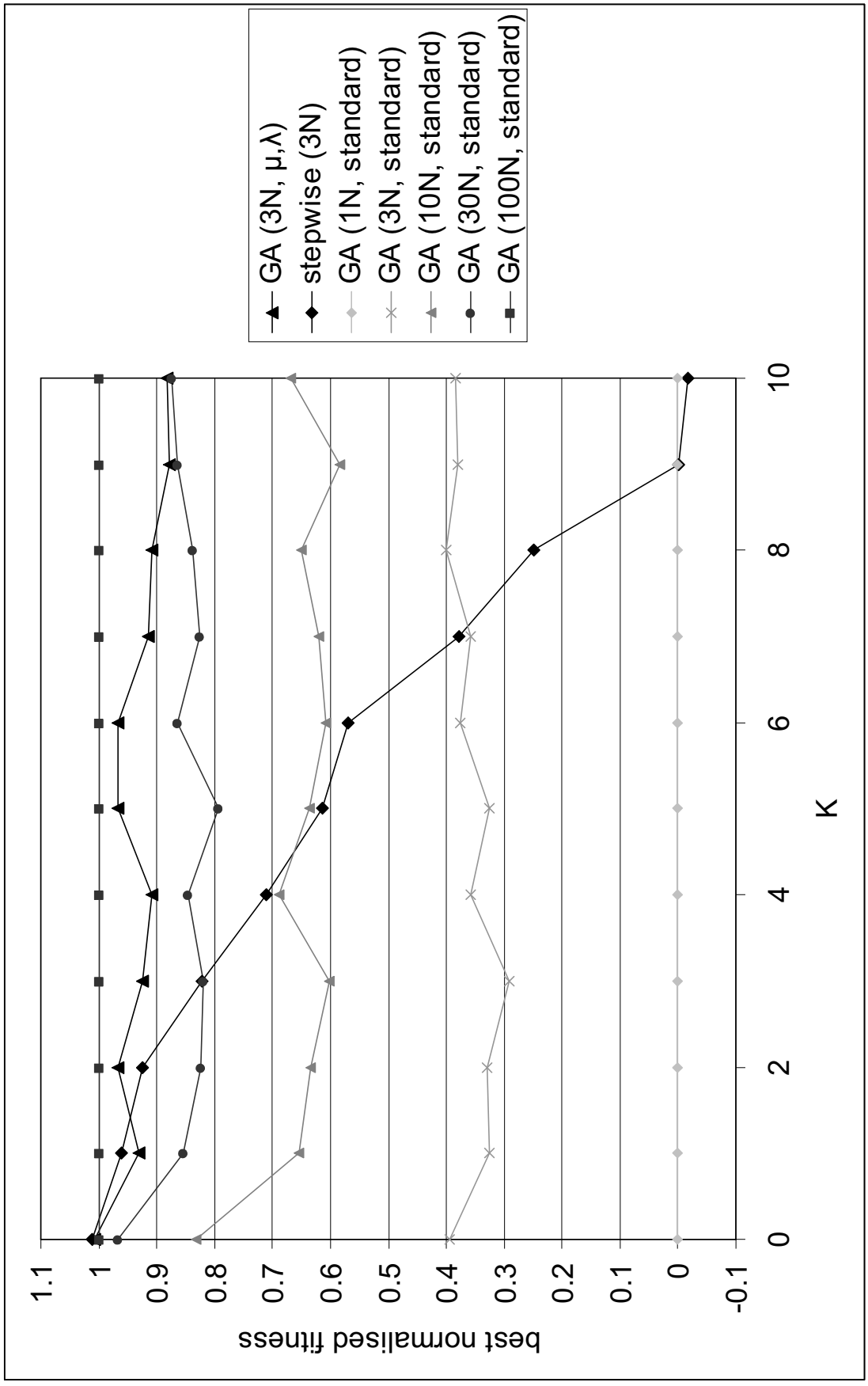

Figure 5