Protein Engineering vol. 16 no. 8 pp. 589–597, 2003
DOI: 10.1093/protein/gzg077

Optimizing the search algorithm for protein engineering by directed evolution

Richard Fox[1,2], Ajoy Roy[1], Sridhar Govindarajan[3], Jeremy Minshull[3], Claes Gustafsson[3], Jennifer T.Jones[4] and Robin Emig[1]

[1]Maxygen, Inc., 200 Penobscot Drive, Redwood City, CA 94063, [3]DNA 2.0, Inc., 1455 Adams Drive, Menlo Park, CA 94025 and [4]Bioren, Inc., 100 Glenn Way, Suite 1, San Carlos, CA 94070, USA

[2]To whom correspondence should be addressed.
E-mail: richard.fox@maxygen.com

An *in silico* protein model based on the Kauffman *NK*-landscape, where *N* is the number of variable positions in a protein and *K* is the degree of coupling between variable positions, was used to compare alternative search strategies for directed evolution. A simple genetic algorithm (GA) was used to model the performance of a standard DNA shuffling protocol. The search effectiveness of the GA was compared to that of a statistical approach called the protein sequence activity relationship (ProSAR) algorithm, which consists of two steps: model building and library design. A number of parameters were investigated and found to be important for the comparison, including the value of *K*, the screening size, the system noise and the number of replicates. The statistical model was found to accurately predict the measured activities for small values of the coupling between amino acids, $K \leq 1$. The ProSAR strategy was found to perform well for small to moderate values of coupling, $0 \leq K \leq 3$, and was found to be robust to noise. Some implications for protein engineering are discussed.

Keywords: directed evolution/genetic algorithm/*in silico*/*NK*-landscape/partial least squares

Introduction

Protein design and optimization is a difficult task, owing in part to the combinatorial explosion of possible molecules that constitute the searchable space. The protein design problem was recently shown to belong to a class of problems known as *NP*-hard (Pierce and Winfree, 2002), indicating that there is no known algorithm that can solve such problems in polynomial time, even if we could obtain a perfect representation *in silico*. Because of this complexity and our limited ability to accurately predict the tertiary structure, let alone *in vivo* activity, of most proteins, many approximate methods have been used to design improved proteins; chief among them is the method of directed evolution. Directed evolution of proteins is today dominated by various high throughput screening and recombination formats, often performed iteratively. Popular examples include DNA shuffling (Stemmer, 1994), synthetic shuffling (Ness *et al*., 2002), multi-gene shuffling (Ness *et al*., 1999), RACHITT (Coco *et al*., 2001), degenerate homoduplex (Coco *et al*., 2002), ITCHY (Ostermeier *et al*., 1999), SCRATCHY (Lutz *et al*., 2001), SHIPREC (Sieber *et al*., 2001), StEP (Zhao *et al*., 1998) and more. These formats are all based on the same general search strategy that guides classical breeding in agriculture and genetic algorithms in computer science.

Sequence space can be described as a space where all possible protein neighbors can be obtained by a series of single point mutations (Smith, 1970). A 100 residue long protein would be a 100 dimensional object with 20 possible values (the 20 naturally occurring amino acids) in each dimension. Conceptually, each one of these proteins has a corresponding fitness on some complex landscape. Models of such 'fitness landscapes' were first studied by Sewall Wright (Wright, 1932) but have since been expanded on by others (Eigen, 1971; Kauffman and Levin, 1987; Kauffman and Weinberger, 1989; Schuster and Stadler, 1994; Govindarajan and Goldstein, 1997).

The sequence space of proteins is immense and is impossible to explore exhaustively. The underlying landscape can be described by the degree of correlation between variables that define the sequence (amino acids, i.e. input) and the function (activity/fitness, i.e. output). In a completely correlated/smooth landscape the partial contribution to fitness of any variable is independent of context, resulting in a 'Mt Fuji' profile. Such fitness landscapes are trivial to optimize by walking up the hill via fitter one mutant neighbors until the global optimum is reached. On the other hand, in a completely uncorrelated/rugged landscape, all the residues are coupled in complex contexts and the landscape is essentially chaotic. In this case no search is more effective than exhaustive sampling of all possible combinations. Simple walks via single point mutation, owing to their high likelihood of getting trapped in local optima, cannot effectively search such rugged landscapes.

Multidimensional optimization strategies that make use of neural nets (NN), genetic programs (GP), self-organizing maps, partial least squares (PLS) and others have been developed during the last 30 years and have been deployed effectively in many disparate fields (Hand *et al*., 2001). The development and application of various data mining and machine learning algorithms is a thriving business today, and is used for everything from gasoline formulation to stock market analysis. These tools and strategies have generally not been applied to protein engineering despite their wide acceptance and commercial success in related fields such as small molecule quantitative structure–activity relationships (QSAR), micro array and single nucleotide polymorphism (SNP) analysis. The exception, to our knowledge, is some work done on peptides as derived from the small molecule QSAR field (Hellberg *et al*., 1986; Jonsson *et al*., 1993; So and Karplus, 1996; Mee *et al*., 1997; Cho *et al*., 1998; Lejon *et al*., 2001) as well as work done on functional protein space mapping by the Husimi group (Aita *et al*., 2000, 2001, 2002).

In order to study the effects of ruggedness, Stuart Kauffman proposed a model of such systems known as the *NK* model (Kauffman and Weinberger, 1989). The *NK* model can be used

Downloaded from http://peds.oxfordjournals.org/ at Aga Khan University on March 10, 2015

to describe any complex landscape, where the complexity is a function of the dependency or coupling of variables that describe the space. The coupling constant, K, is somewhere between 0 (completely correlated) and $N - 1$ (completely uncorrelated), where N is the number of variable positions. From a protein engineering standpoint, K is equal to the average number of variable positions functionally coupled to another variable position.

Protein engineering through directed evolution is nicely positioned to take advantage of modern, multidimensional search algorithms:

1. Protein sequence space is defined. Each residue position (X variable) in the linear protein polypeptide corresponds to one of the 20 natural amino acids. The X variables are discrete and atomic.
2. Protein sequence space is essentially infinite. Even in a small protein of 100 residues with 20 options for each residue, the total space is 10^{130} (20^{100}) unique sequence combinations. This number is sufficiently large to equate with infinity for all practical purposes. However, protein optimization searches are usually limited to a much smaller, though still vast, set of potential sequences. Only the variable portions of the sequence are part of the search strategy.
3. There is a direct mapping of sequence (X) to function (Y). All the phenotypically controlling information is encoded either directly or indirectly in the amino acid sequence of the protein, given a fixed expression and assay environment. In this case only changes in the amino acid variables X are manifested in the function of the protein Y.
4. Information (nucleic acid sequence) is physically linked to function (protein) as they reside in the same cell, same compartment or are chemically linked. This co-location facilitates iterative identification and recombination of X variables.
5. DNA shuffling and other sexual recombination formats generate systematically varied data sets. Having such data sets is necessary in order to establish accurate X–Y correlations.
6. The advent of high throughput DNA sequencing is making the acquisition of accurate sequences fast and inexpensive.

In this paper we use an *in silico* fitness simulation based on the Kauffman *NK*-landscape in order to compare different search algorithms as applied to protein engineering.

DNA shuffling as a genetic algorithm

The genetic algorithm (GA) that underlies the principle of evolution by natural selection was investigated theoretically by John Holland and is typically applied to optimization problems that have many input variables and a complex function to optimize (Holland, 1975). The canonical GA consists of the following steps:

1. Screen a small number of samples from a combinatorial library.
2. Select more active parents for mating and breeding
3. Build the next round library by mating the parents, allowing for recombination of parent genotypes
4. Iterate steps 1 through 3 to achieve higher fitness.

Holland showed that the GA is both efficient and robust for a wide class of optimization problems. Consequently, the method has become very popular for searching the multidimensional space of many diverse problems. Genetic algorithms have proven themselves worthy in solving countless optimization problems and are routinely applied now to applications ranging from manufacturing optimization to financial decision-making.

DNA shuffling is analogous to a GA done in a wet laboratory environment but with at least two important differences. (i) Typically, the *in silico* GA has a scoring function or fitness function that is exact and continuous throughout the entire search space. For directed evolution in general, the *in vitro* or *in vivo* fitness function is inexact due to experimental error and only available for individual sequences having a fitness above an experimental cut-off. (ii) A typical *in silico* GA is performed over dozens or hundreds of generations whereas DNA shuffling is rarely performed over >10 generations.

Protein sequence activity relationship algorithm

Here, we introduce a potentially more efficient search algorithm predicated on the advent of high throughput sequencing and modern statistical techniques and compare the new algorithm with the standard GA described above. The protein sequence activity relationship algorithm (ProSAR) algorithm used in this work consists of the following steps:

1. Screen a small number of samples from a combinatorial library, associating sample activities with their sequences.
2. Build a statistical model that correlates the sequences with the measured activities.
3. Use the statistical model to identify residues that contribute most to improving the desired activity or fitness.
4. Build the next round library by incorporating residues that contribute most to the predicted fitness, rejecting those residues predicted to confer low fitness. Residues with putative neutral contributions in the current combinatorial context may be kept in the search in order to find potentially beneficial epistatic interactions in the new context.

Iterate steps 1 through 4 to achieve higher fitness.

Materials and methods

NK-landscape

Just as in other fields of engineering, having an *in silico* simulation of a system allows for immediate testing and validation of alternative strategies. The simulation used here is generic to protein sequence-fitness correlations and is useful for exploring how to best navigate through a complex landscape. The *NK*-landscape used in this study is not meant to simulate any real protein of interest. Rather it affords us the ability to study the power of various search algorithms on tunably rugged landscapes. Often the ruggedness of a protein landscape has been looked at in terms of how the constituent residues of a protein deviate from the principle of additivity. Wells studied the additivity of mutational effects for transition-state stabilization energies (Wells, 1990). When the variable residues were not close enough to contact each other, the fitness landscape was not rugged. Others have looked at various protein properties and protein–protein interactions and found that the corresponding fitness landscapes were not rugged (Sandberg and Terwilliger, 1993; Nikolova *et al.*, 1998; Weiss *et al.*, 2000; Lu *et al.*, 2001; Benos *et al.*, 2002; Vajdos *et al.*, 2002). Conversely, rugged landscapes may exist for various problems of interest in protein engineering (Dill, 1997).

A sample protein sequence and *NK*-landscape for $N = 10$ and $K = 1$ are shown in Figure 1. The sequence itself is expressed as a set of bits (0 or 1), representing the choice between two different amino acids at a given position. Only the variable

Downloaded from http://peds.oxfordjournals.org/ at Aga Khan University on March 10, 2015

Variable Position	1	2	3	4	5	6	7	8	9	10
Choice 0	I	L	L	M	G	W	K	C	S	F
Choice 1	V	A	I	P	H	N	R	T	A	Y
Sequence Value	V	A	L	P	G	W	K	T	S	F
Coded Value	1	1	0	1	0	0	0	1	0	0

1	2	w_1
0	0	0.9
0	1	0.1
1	0	0.3
1	1	0.4

. . .

9	10	w_9
0	0	0.1
0	1	0.9
1	0	0.5
1	1	0.5

2	3	w_2
0	0	0.5
0	1	0.2
1	0	0.8
1	1	0.1

3	4	w_3
0	0	0.7
0	1	0.3
1	0	0.7
1	1	0.5

Fig. 1. A sample protein sequence and *NK*-simulated protein fitness landscape for $N = 10$ and $K = 1$. The protein sequence is coded in a binary form with bit value 1 representing one possible residue choice and bit value 0 representing another residue choice at each variable position. The *NK*-landscape's look-up tables contain randomly generated values and are shown with the current values shaded that correspond to the sample protein sequence. Each random value represents the partial contribution to fitness, w_i, at that variable position. In this example, with $K = 1$, each residue's partial contribution to fitness is functionally coupled to one other variable residue in the protein.

positions are coded in the sequence and there is no expressed or implied proximity of the variable positions. All the modeling and representations that follow are independent of the variable ordering. For coding simplicity, coupling between residues is introduced via logical adjacency, but the real coupling could easily be envisioned as distant. For example, the model shown in Figure 1 shows coupling between variable positions 1 and 2, but it could have also been coded to create coupling between variable positions 1 and 6. In the GA strategy described below, which is more akin to synthetic shuffling, we do not preserve 'motifs' like classical DNA shuffling, so the order and position of any coupling is immaterial.

The sequence shown in Figure 1 may look unfamiliar to a protein engineer, but any protein alignment that consists of two options at a given variable position can be cast as a matrix of bits. The bit matrix can be generalized to include more than two options at each variable position but, for the sake of simplicity, we only consider two option models here. Adding more options at each variable position does not change the underlying logic of the model.

Physico-chemical descriptors are often used in favor of amino acid symbols when building the input matrix, however, the use of quantitative molecular descriptors is only appropriate when enough of them are available to build a robust, continuous input variable. When only two residues are available at a given variable position, a binary representation can capture the available information. The choice of using a 0 or 1 bit to represent a specific residue is immaterial since the statistical model will flip the sign for the fitness contribution depending on whether the residue is advantageous or deleterious.

At every variable position in the protein, the partial fitness, w_i, is drawn from a look-up table generated randomly when the landscape is first created. The look-up table is used to map portions of the sequence to a continuous random variable. In the example shown in Figure 1, the part of the sequence that contains positions 2 and 3 are used to determine the value for the partial fitness at position 2 (positions 2 and 3 are coupled when $K = 1$).

The total fitness of the protein is then given by summing the partial fitness values over each variable position:

$$W = \frac{100}{N}\sum_{i=1}^{N} w_i \qquad (1)$$

In the original *NK* landscape proposed by Kauffman, the random variables were drawn from a uniform distribution between 0 and 1 (Kauffman and Weinberger, 1989). For most of the landscapes attributes he studied, the underlying distribution of the random variables was not important since those attributes (like the number and density of local optima) depend only on rank order rather than the numerical fitness values (Kauffman, 1993). For this study, in order to simulate what is often observed in actual proteins, we have chosen to draw the random variables used in the look-up tables from a gamma distribution with a mean and variance of 0.35. With this distribution, more residues tend to behave in a neutral fashion (their contribution to fitness is small) and a few residues tend to have a relatively large impact on fitness.

Genetic algorithm details

The GA used to search the *NK* landscape is intended to capture most of the salient features of DNA shuffling and hereafter the GA and DNA shuffling strategies will be considered equivalent. The GA works, as described in general in the Introduction, with the following specific rules:

1. A subset of random protein sequences is generated from the combinatorial library of possible sequences.
2. The proteins are ordered from highest to lowest fitness.
3. A number of high fitness proteins are selected from the top of the list using the following algorithm:
   a. The top protein is retained as the backbone.
   b. Subsequent proteins are then added in the order of decreasing fitness, and their residues are added to the list of residues at each position in the backbone.
   c. The effective library size is calculated for every new residue added to the mix. The effective library size can be thought of as the inverse of the probability of selecting the same sequence on two random draws from the library. The effective library size, L, is calculated by looking at the biases of each residue at a given position:

$$L = \prod_{i=1}^{M} \frac{1}{\left(b_i^2 + (1 - b_i)^2\right)} \qquad (2)$$

where b_i is the bias (frequency) in the amino acid at position i. This is essentially a measure of the information content contained at each position. The above reduces to $L = 2^M$ when $b_i = 0.5$ at every position (maximal information content). As the biases deviate

Downloaded from http://peds.oxfordjournals.org/ at Aga Khan University on March 10, 2015

from 0.5, the effective library size decreases (the information content decreases).

d. The calculated library size is then compared to the desired library size. If the library size is greater than or equal to the desired library, we create a new library based on the current set of best proteins. Diversity is quenched from the system in subsequent rounds of selection and reproduction. Formulating the selection this way allows us to precisely control the library size at each round for the GA strategy (in order to match the library size used in the ProSAR strategy)—a necessary feature for fair comparisons between the search algorithms. Otherwise, greedy over-selection in either strategy can lead to increased fitness at the expense of prematurely exhausting diversity.
e. Progeny proteins are then randomly created using the residue biases found in the top proteins. Again this strategy is similar to synthetic shuffling, where the probability of incorporating a given residue is independent of the residues incorporated at other, nearby positions.

4. The new proteins are ranked by fitness, and steps 2 and 3 are repeated.

ProSAR details

Modeling step. The modeling step of ProSAR is based on the PLS variable regression technique. PLS is a predictive technique that can handle many independent variables, even when these variables display multicollinearity. It is preferable for use in problems with many independent variables and few observations. Though developed by Herman Wold (Wold, 1981, 1985) for econometrics, PLS gained popularity in chemometrics research and later in industrial applications. It has since spread to research in education, marketing and the social sciences. In biology, PLS is well-established in QSAR studies of small molecules and peptides.

PLS uses *X* variable importance regression to build predictive models based on multicollinearity among variables and their correlation with *Y*-score. The *X*- and *Y*-scores are selected by PLS so that the relationship of successive pairs of *X*- and *Y*-scores is as strong as possible (Hand *et al*., 2001).

The advantages of PLS include its ability to model multiple dependent as well as multiple independent variables; the ability to handle multicollinearity among the independents; robustness in the face of data noise and missing data; all making for stronger predictions. Consequently, PLS is often favored as a predictive technique and less as an interpretive technique. Parallel analysis of NN, genetic programming and decision trees give similar results.

The details of how PLS works to arrive at the final regression equation can be found elsewhere (Geladi and Kowalski, 1986). The upshot of PLS modeling can be written in the form of a predicted response (regression) equation:

$$y = c_0 + c_1x_1 + c_2x_2 + c_3x_3 + \ldots + c_Nx_n = \sum_{0}^{N} c_ix_i \qquad (3)$$

where y is the predicted response (fitness), c_i and x_i are the regression coefficient and bit value (residue choice) respectively at position i. The form of Equation 3 assumes a linear (additive) relationship between the residues at each position. Higher order models that include cross product terms may be able to explicitly capture interactions between residues but generally requires more training samples. Feeding too few samples into a regression having many non-linear terms may result in over fitting the model to the data. In such cases, the model can essentially memorize the training set, but then displays very poor predictive capabilities when presented with new, unseen samples. The less sensitive nature of the pure linear model may actually be an asset when too few samples are available to build robust non-linear models.

Equation 3 encodes the choice of two residues at one position as a choice between the values $x_i = 0$ or $x_i = 1$. If we have more than two possible residues at one position, Equation 3 can be rewritten with each variable (x_1, x_2, x_3, etc.) representing the presence or absence of a given residue rather than a choice between two residues. For example, if the amino acids Ile, Val and Leu are all possible at one position then the variables would take on the following values: when Ile is present $x_1 = 1$, $x_2 = 0$, $x_3 = 0$; when Val is present $x_1 = 0$, $x_2 = 1$, $x_3 = 0$; when Leu is present $x_1 = 0$, $x_2 = 0$, $x_3 = 1$. PLS treats either formulation equivalently. The quality and results of the model are unchanged. In general, the sum of the regression coefficients equals zero for a given position when formulated this way. This presence/absence coding scheme can also be used to encode any number of possible residues at a given position.

The ability to take advantage of such linear regression depends on the additivity of the functional contribution from each residue. If the fitness contribution from each amino acid residue in the protein is additive, there will be no functional coupling and *K* of the *NK*-landscape will be 0. As *K* increases, the ability to model the sequence-function space decreases. By computationally generating a random function given the *NK*-landscape parameters, we can look at the consistency of the model for different values of *K*. The measured versus predicted values of fitness for four different values for *K* are plotted in Figure 1 for $N = 40$ variable positions (the total number of different sequences in this space = $2^{40} = 10^{12}$). The plots shown in Figure 1 were done using 'leave one out' cross-validation. The models were built using $3N$ (120) random data points, so for each of the 120 data points, a model was built using the other 119 data points to train the model and the left out data point was then predicted by the model. As shown in Figure 2a, measuring the fitness of 120 systematically varied sequences is enough to reliably predict fitness values for all 10^{12} possible sequences, given $K = 0$. PLS is accordingly well suited to deconvolute the independent contributions from the different residues in a $K = 0$ landscape. As *K* increases, the qualitative value of the model decreases and at $K = 3$ couplings per residue, the ability of the model to accurately predict the measured fitness is reduced. However, as we will show later, this inability to accurately predict the fitness at higher coupling does not mean there is no practical information contained in the model. In fact we can extract useful information to make fitter libraries in the next round.

The PLS model has a tendency to under-predict the response at the high end and over-predict the response at the low end when $K > 0$, owing to the modeling of an inherently non-linear system with a linear equation.

Library design step. The PLS model could be used for protein optimization if the relative influence of each residue can be deconvoluted from the others and the residue tendencies of an

Downloaded from http://peds.oxfordjournals.org/ at Aga Khan University on March 10, 2015

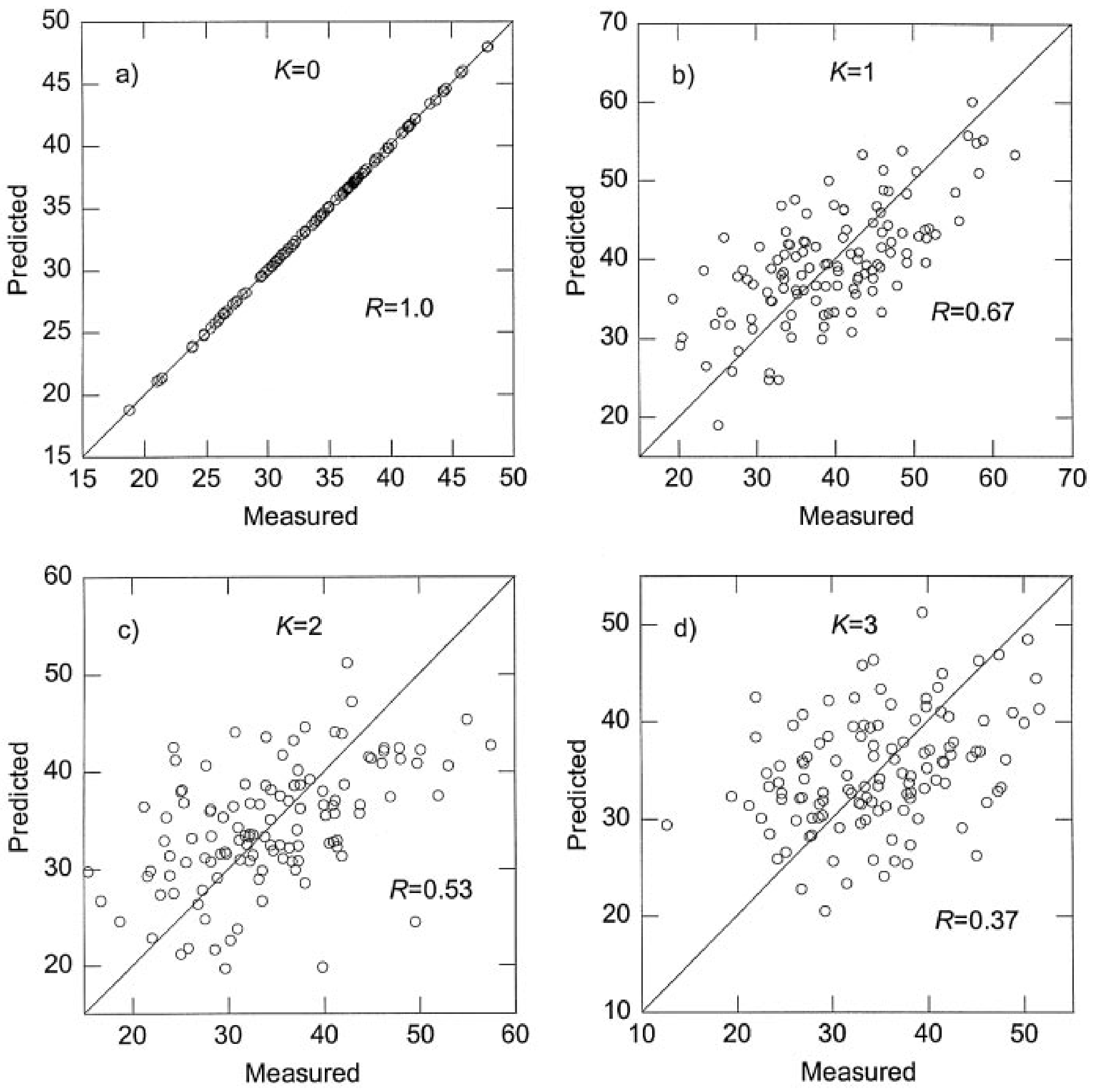

Fig. 2. The effect of *K* on modeling. (**a**) through (**d**) display the predicted fitness values of the model compared with measured fitness values for increasing *K* using leave-one-out cross validation. The abscissa is the measured value, computed using the *NK*-simulated landscape. The ordinate is the corresponding predicted value extracted from the PLS model. The solid line in each panel has a slope of 1. *R* is the correlation co-efficient, a measure of the association between predicted and measured values, ranging from 1 (perfect correlation) to 0 (no correlation).

improved sequence deduced. To this end, a sample plot of the regression co-efficients obtained from PLS modeling is shown in Figure 3. In this example, the residue choice at position 7 appears to have the most influence on the model's predicted fitness. Because the regression coefficient is negative at position 7, the residue encoded by the 0 bit is the superior choice. The next most important residue choice is found at position 14, with the 1 bit encoding the superior residue since the regression coefficient is positive at this variable position.

Residues with large regression co-efficients are those found to be beneficial in many contexts. Although we use a linear form of the PLS model in this work, we are not limited to identifying beneficial residues that are acting in strictly additive ways. A large, positive regression co-efficient might indicate that the particular residue encoded by the 1 bit, though strongly coupled to other residues, exhibits generally positive contributions to fitness in many contexts. This is a powerful concept that explains much of the search prowess achieved with the ProSAR strategy.

The regression co-efficients are used to construct a library for the next round that has improved fitness. We do this by fixing the most important residues to a value that maximizes the predicted fitness while retaining variation at other positions. For example, if we started with $N = 40$ variable positions in the first round library, in the second round library we could fix the top 20 most important residues as measured by their regression coefficients and vary the remaining 20 positions that have smaller regression coefficients.

The ProSAR optimization strategy is used to accelerate the selection of beneficial residues. Analogous to genes evolving in nature, where the allele frequency dictates the mean fitness of organisms and the heritable unit is the 'selfish gene' (Dawkins, 1992), in ProSAR the residue frequency dictates the mean fitness of proteins and the heritable unit is the 'selfish residue'.

The fact that a given position has a low regression co-efficient in one round does not necessarily mean that the choice of residues at that position is not ultimately important. Fixing a high value regression co-efficient residue in one round may

Downloaded from http://peds.oxfordjournals.org/ at Aga Khan University on March 10, 2015

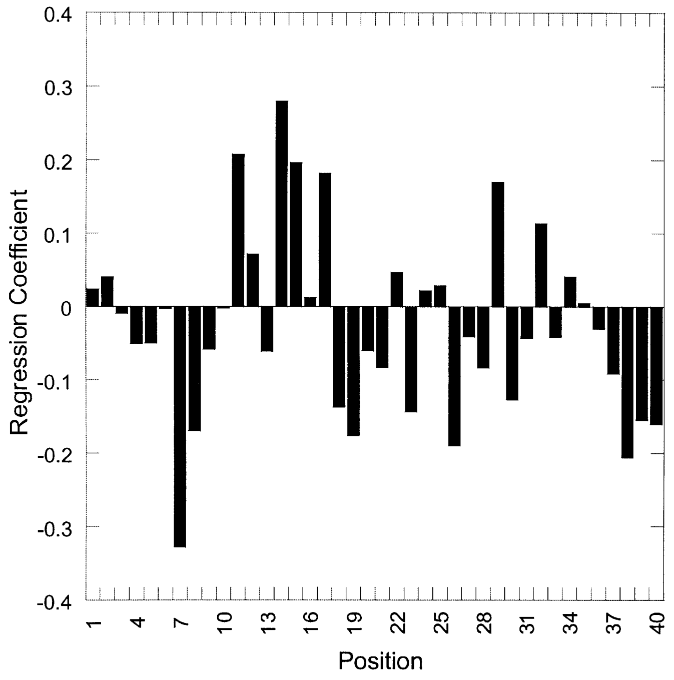

Fig. 3. PLS regression co-efficients from an *NK*-simulated landscape with $N = 40$, $K = 1$ and screened with $3N$ data points. The abscissa is the variable position number. The ordinate is the value of the regression co-efficient, c_i, for that position. Positive values of the regression coefficient indicate the residue corresponding to bit value 1 confers higher fitness; negative values indicate the residue corresponding to bit value 0 confers higher fitness. In this example, the residue corresponding to a bit value of 0 at variable position 7 is the most important residue conferring higher fitness.

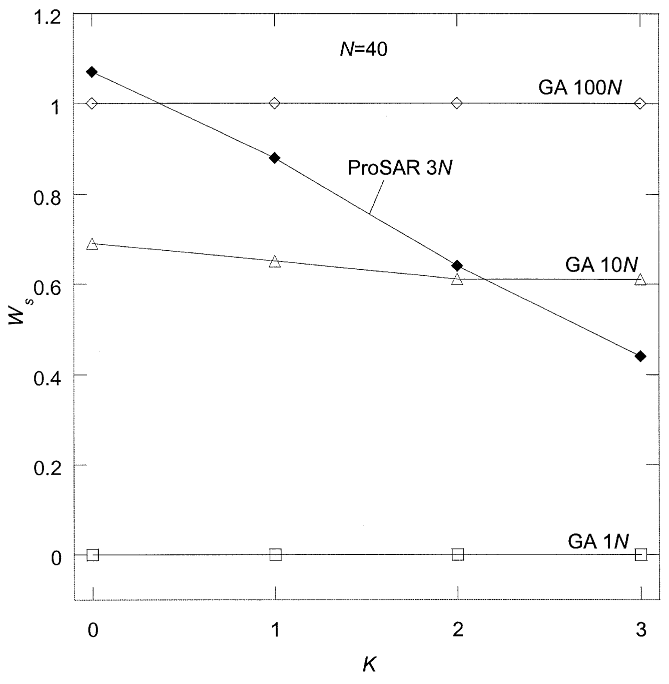

Fig. 4. Comparison of the ProSAR and GA strategies with increasing values of K for an *NK*-simulated landscape with $N = 40$ variable positions. The abscissa is the epistatic coupling parameter, K, between variable positions. The ordinate is the best scaled fitness, W_s, achieved after the second round of evolution and averaged over 300 randomly generated landscapes. Each curve shows the results using different methods and/or screening sizes. For example, the GA $100N$ curve shows the fitness achieved using the GA strategy and screening $100N$ or 4000 samples per round. Alternatively, the ProSAR $3N$ curve shows the fitness achieved using the ProSAR strategy after screening $3N$ or 120 samples per round.

allow other residues to flourish in the new context (Spiller *et al.*, 1999).

Results

ProSAR versus GA with increasing K

In order to compare the efficiency of the ProSAR optimization strategy with that of the standard GA, we calculated the average fitness value for the best protein at the end of the second round for 300 randomly generated landscapes. In each case studied here we started with an initial diversity $N = 40$ and quenched the diversity in the second round to $N = 20$. Although most directed evolution programs typically go through more iterations, we wanted to investigate differences between the two algorithms for a single step in the process.

In order to compare the alternative optimization strategies, we studied the two approaches for different values of the coupling parameter K. The scaled fitness values were calculated as follows:

$$W_s = \frac{W - W(GA, 1N)}{W(GA, 100N) - W(GA, 1N)} \tag{4}$$

where W_s is the scaled fitness value, W the measured fitness value and $W(GA,1N)$ and $W(GA,100N)$ the measured fitness values from the GA at screening sizes $1N$ and $100N$, respectively. The fitness values were scaled in order to compare the results between different values of K since the peak fitness values tend to vary with K, a known artifact of the *NK*-landscape (Kauffman, 1993). Figure 4 shows a plot of the scaled fitness values for both the GA and ProSAR strategies.

Figure 4 shows how the ProSAR algorithm at screening size $3N$ compares with different GA screening sizes. For small values of K, ProSAR compares well with the GA. At $K = 0$, using the ProSAR strategy and sampling $3N$ data points achieves a higher fitness value than the GA strategy sampling $100N$ data points. A screening size of $200N$ for the GA strategy was required to achieve the same fitness using ProSAR, resulting in over a 66-fold increase in screening efficiency for ProSAR. This result is not unexpected since the PLS model is nearly perfect in terms of its ability to estimate the regression co-efficients and select for the best residue choices. When $K = 1$, the ProSAR $3N$ strategy was found to be over 16-fold more efficient in terms of the screening size required to achieve the same fitness as the GA strategy. At increasing values of K, the efficiency of the ProSAR strategy decreases, though it still performs well compared to the GA $10N$ strategy when $K = 2$, an ~3.3 fold increase in screening efficiency. In this relatively rugged landscape, where every variable position is functionally coupled to two other variable positions, the linear PLS model has a greater difficulty identifying residues that are beneficial in many contexts.

For small to moderate values of the coupling K, ProSAR appears to be a useful strategy to find sequences of higher fitness at a fraction of the screening cost required for the GA strategy.

Downloaded from http://peds.oxfordjournals.org/ at Aga Khan University on March 10, 2015

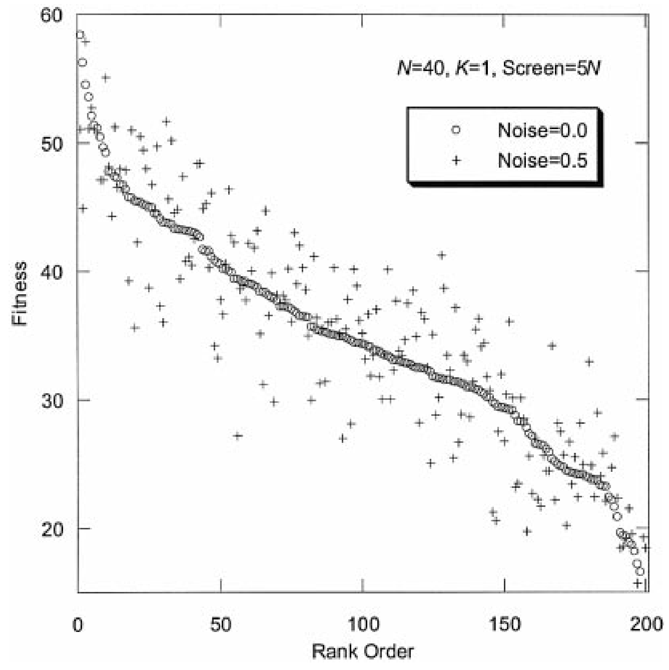

Fig. 5. A plot of the fitness values before and after the addition of noise based on an *NK*-landscape with $N = 40$, $K = 1$ and a screening size of 5*N* samples. The abscissa is the rank order of fitness values based on the pristine (unperturbed) fitness value. The ordinate is the fitness value before and after the addition of noise. Applying a random Gaussian error whose standard deviation is 50% (Noise = 0.5) of the standard deviation of the pristine values generates the perturbed fitness values.

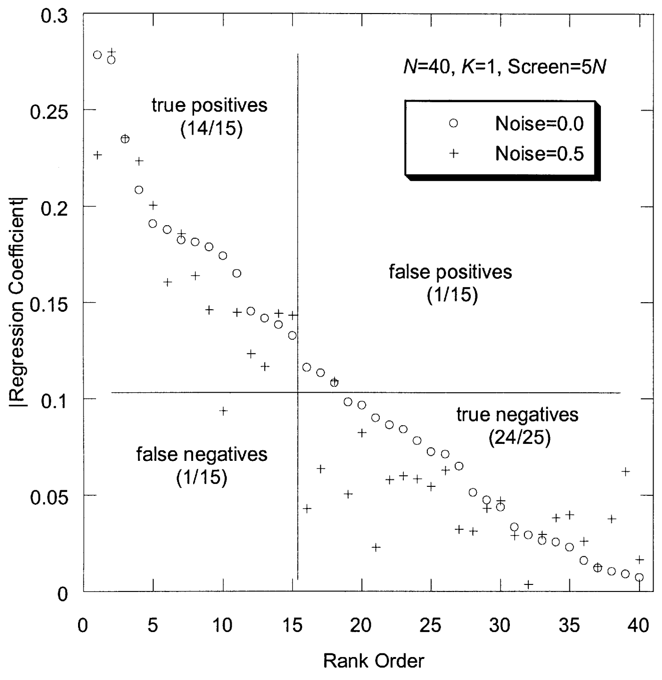

Fig. 6. A plot of the absolute value of the regression co-efficients before and after the addition of noise based on an *NK*-landscape with $N = 40$, $K = 1$ and a screening size of 5*N* or 200 samples. The abscissa is the rank order of the absolute value of the regression coefficient before adding noise. The ordinate is the absolute value of the regression coefficients before and after the addition of noise. The regression coefficients were calculated based on the same data set used to plot Figure 5.

ProSAR versus GA with increasing noise

We also wanted to consider the impact that experimental noise has on the GA and ProSAR strategies. In order to model the noise one might see in the laboratory, it is not, in general, sufficient to specify just the percent error without considering the dynamic range of the assay. For example, a 25% error for an assay that spans a factor of two in fitness is very different from a 25% error that spans an order of magnitude in fitness; the latter contains much less fractional error. Accordingly, we constructed a noise model using the following equation:

$$W_{\text{noise}} = W + n\sigma_W G \qquad (5)$$

where W_{noise} is the measured fitness value after adding noise, W the pristine fitness value calculated directly from the *NK*-simulated landscape, n the noise level, σ_W the standard deviation of all measured fitness values and G a random variable drawn from a Gaussian distribution that has a standard deviation of one. An example of applying the above equation to a sample data set is shown Figure 5. Two hundred data points are plotted in rank order of their pristine fitness values and labeled as $n = 0.0$. Then, the same data set is perturbed using the above equation with an $n = 0.50$ noise level.

If we were to re-rank the data points after applying noise, we would likely see a significant number of data points with high pristine fitness values land further down in the distribution. Conversely, some data points with lower pristine fitness values would land at the top of the distribution. This reordering of data points near the top of the distribution would clearly have an impact on the standard GA because it operates on the basis of ranking data points.

What is the impact of noise on the order of the regression coefficients? We can answer this question by plotting the absolute value of the regression coefficients both before and after the addition of 50% noise for a model based on sampling 5*N* data points (Figure 6). After adding noise, the PLS model still accurately predicts 14 of the top 15 residues. Only one false positive and one false negative were predicted. Likewise, 24 of 25 true negatives were predicted correctly.

The above provides a qualitative understanding of how robust the ProSAR strategy might be to noise, but a direct comparison of the GA and ProSAR strategies with varying noise levels is more informative. A linear increase in the screening size for ProSAR strategy compares well with an order of magnitude increase in screening size for the GA strategy at all noise levels (Figure 7). Furthermore, the ProSAR strategy appears to be relatively robust to noise. For example, the fitness value achieved via the GA 100*N* strategy is somewhat higher than that achieved using the ProSAR 3*N* strategy when $n = 0.0$, but at a higher noise, $n = 1.5$, the GA 100*N* strategy drops off to equal the ProSAR 3*N* strategy.

Another hidden advantage of the ProSAR strategy may be found in noting that experimental protocols using high throughput assays often have higher noise or are not as commercially relevant as surrogates for the low throughput target assays. In all the cases studied, ProSAR was found to be an efficient strategy for finding equivalent or superior fitness values while screening far fewer samples, anywhere from 10 to 100 times more efficient. Such relatively low screening

Downloaded from http://peds.oxfordjournals.org/ at Aga Khan University on March 10, 2015

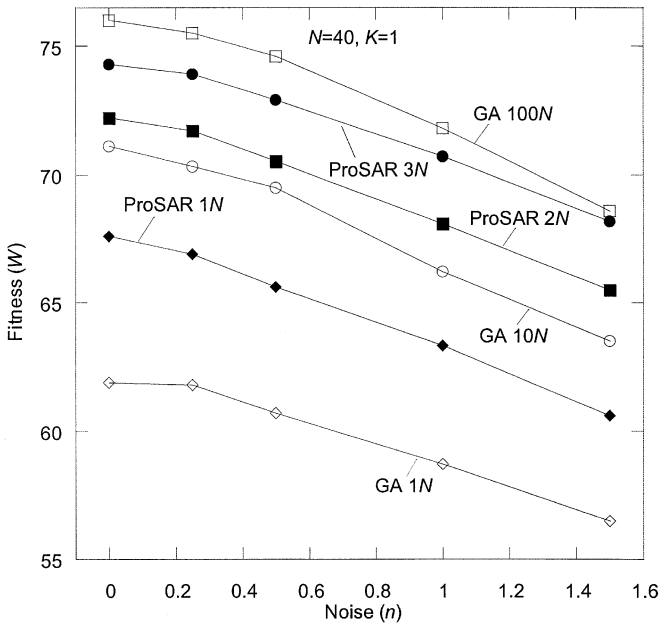

Fig. 7. Comparison of the ProSAR and GA strategies for varying levels of noise. The curves represent the fitness achieved after evolution on an *NK*-landscape with $N = 40$ and $K = 1$. The abscissa is the amount of noise added to the samples. The ordinate is the fitness value obtained using each strategy and screening size.

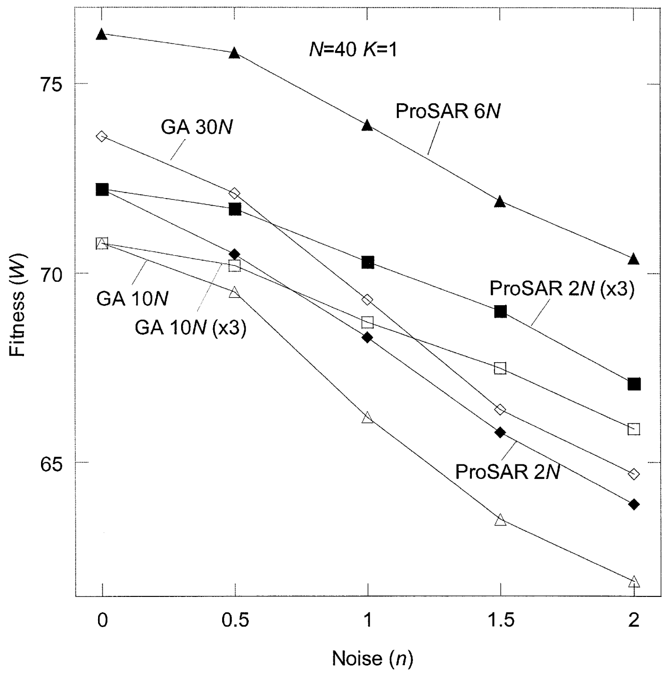

Fig. 8. The effect of replicates for varying levels of noise. The curves represent the fitness values achieved after evolution on an *NK*-landscape with $N = 40$ and $K = 1$. The abscissa is the amount of noise added to the samples. The ordinate is the fitness value obtained using each method and screening size. Curve labels that contain multiplication factors indicate they were generated using replicates. For example, the label ProSAR 2*N* (×3) means the curve was generated using the ProSAR method, screening 2*N* unique sequences in triplicate. Alternatively, those curves without multiplication factors indicate no replicates were performed. For example, the GA 30*N* label means the curve was generated using the GA method, screening 30*N* unique sequences.

requirements for ProSAR means that it is a potentially useful strategy for use in high quality, low throughput assays.

Effect of replicates

One might reasonably argue that doing replicate assays at higher noise levels could help reduce the error associated with each data point and improve the search prowess of either strategy, despite the higher cost in time and money to screen more data. In order to investigate such an approach, we calculated the effect of doing replicates for both the GA and ProSAR strategies. The results are shown in Figure 8. The initial GA 10*N* curve is similar to that shown in Figure 7, but taken out to a higher noise level. For the GA screen using triplicates, GA 10*N* (×3), at noise $n = 0.0$, the fitness value is the same without triplicates; this is expected since there was no error in the original 10*N* screen and doing replicates was a waste of effort. At higher noise levels, the curve using triplicates for the GA tends to be more robust to noise than the GA 10*N* curve without triplicates. The total number of samples screened at 10*N* with triplicates is actually 30*N*. Thus, we can then ask if it is better to put the increased screening effort used to do replicates into simply screening more unique sequences, each with higher error. For the GA strategy, the answer to the question depends on the amount of noise in the assay. The fitness values at low noise and GA 30*N* are superior to those at GA 10*N*. However, at higher noise levels, the GA 10*N* strategy with triplicates outperforms the GA 30*N* strategy.

The ProSAR 2*N* strategy shown in Figure 8 is also the same as the one seen in Figure 7. However, if we allow for triplicates, PLS 2*N* (×3), we get an improved (more robust) curve at higher noise levels. Interestingly, if we put the same screening effort as ProSAR 2*N* (×3) into building a PLS model based on 6*N* unique sequences (each with higher noise), ProSAR 6*N*, we get a curve that shows improved fitness values at all noise levels. In other words, it seems that for the ProSAR strategy it is always better to screen more unique sequences (even though each one has a higher error) than it is to put forth the equivalent screening effort to do replicates. The ProSAR strategy appears to handle unique, noisy data better than does the GA strategy. The robustness of the ProSAR algorithm is due mainly to the fact that the modeling uses all the available information in the data set to isolate the important residues and only those that are consistently beneficial in many contexts are selected for, thus averaging out noise effects more effectively.

Discussion

The *NK*-landscape used in this paper simulates the sequence–fitness relationship of a tunably rugged protein landscape. The ruggedness of such landscapes in real protein engineering efforts appears to vary over the range $0 < K < 3$ (Aita *et al.*, 2001, 2002; Govindarajan *et al.*, 2003). It has been argued that it is implicit from evolution that K must be low (Kauffman, 2000). A high K would make the recombination based search of sequence space slow and evolution would be less robust to changes in the environment (Taverna and Goldstein, 2002).

If the *NK*-landscape used in this study is a realistic simulation of actual proteins, then the results presented here indicate that the ProSAR algorithm is a promising search strategy for use in directed evolution. Its potential strengths include the ability to achieve higher fitness values with a

Downloaded from http://peds.oxfordjournals.org/ at Aga Khan University on March 10, 2015

relatively small number of samples and the ability to handle noisy data robustly.

Experimental validation of the method is possible using the algorithm described above. Classical or synthetic DNA shuffling along with rapid, high throughput sequencing can be used to generate the input data sets for use in building the PLS model. The best residues predicted by the model can be incorporated into new sequences by site-directed mutagenesis, or splicing by overlap extension. The new sequences can then be tested to confirm if improved fitness is conferred by the presence of the best predicted residues. It should be possible to compare directly the ProSAR and GA strategies described here in the laboratory.

References

Aita,T., Urata,S. and Husimi,Y. (2000) *J. Mol. Evol.*, **50**, 313–323.

Aita,T., Iwakura,M. and Husimi,Y. (2001) *Protein Eng.*, **14**, 633–638.

Aita,T., Hamamatsu,N., Nomiya,Y., Uchiyama,H., Shibanak,Y. and Husimi,Y. (2002) *Biopolymers*, **64**, 95–105.

Benos,P.V., Bulyk,M.L. and Stormo,G.D. (2002) *Nucleic Acids Res.*, **30**, 4442–4451.

Cho,S.J., Zheng,W. and Tropsha,A. (1998) *J. Chem. Inf. Comp. Sci.*, **38**, 259–268.

Coco,W.M., Levinson,W.E., Crist,M.J., Hektor,H.J., Darzins,A., Pienkos,P.T., Squires,C.H. and Monticello,D.J. (2001) *Nat. Biotechnol.*, **19**, 354–359.

Coco,W.M., Encell,L.P., Levinson,W.E., Crist,M.J., Loomis,A.K., Licato,L.L., Arensdorf,J.J., Sica,N., Pienkos,P.T. and Monticello,D.J. (2002) *Nat. Biotechnol.*, **20**, 1246–1250.

Dawkins,R. (1992) *Selfish Gene*. Oxford University Press, New York, NY.

Dill,K.A. (1997) *J. Biol. Chem.*, **272**, 701–704.

Eigen,M. (1971) *Naturwissenschaften*, **58**, 465–523.

Geladi,P. and Kowalski,B. (1986) *Anal. Chim. Acta*, **198**, 1–17.

Govindarajan,S. and Goldstein,R.A. (1997) *Proteins*, **29**, 461–466.

Govindarajan,S., Ness,J.E., Kim,S., Mundorff,E.C., Minshull,J. and Gustafsson,C. (2003) *J. Mol. Biol.*, **328**, 1061–1069.

Hand,D.J., Mannila,H. and Smyth,P. (2001) *Principles of Data Mining (Adaptive Computation and Machine Learning)*. MIT Press, Boston, MA.

Hellberg,S., Sjöström,M. and Wold,S. (1986) *Acta Chem. Scand. B*, **40**, 135–140.

Holland,J.H. (1975) *Adaption in Natural and Artificial Systems*. University of Michigan, Ann Arbor, MI.

Jonsson,J., Norberg,T., Carlsson,L., Gustafsson,C. and Wold,S. (1993) *Nucleic Acids Res.*, **21**, 733–739.

Kauffman,S. (1993) *The Origins of Order*. Oxford University Press, New York, NY.

Kauffman,S. (2000) *Investigations*. Oxford University Press, New York, NY, pp. 18–19.

Kauffman,S. and Levin,S. (1987) *J. Theor. Biol.*, **128**, 11–45.

Kauffman,S.A. and Weinberger,E.D. (1989) *J. Theor. Biol.*, **141**, 211–245.

Lejon,T., Strom,M.B. and Svendsen,J.S. (2001) *J. Peptide Sci.*, **7**, 74–81.

Lu,S.M. *et al*. (2001) *Proc. Natl Acad. Sci. USA*, **98**, 1410–1415.

Lutz,S., Ostermeier,M., Moore,G.L., Maranas,C.D. and Benkovic,S.J. (2001) *Proc. Natl Acad. Sci. USA*, **98**, 11248–11253.

Mee,R.P., Auton.T.R. and Morgan,P.J. (1997) *J. Peptide Res.*, **49**, 89–102.

Ness,J.E., Welch,M., Giver,L., Bueno,M., Cherry,J.R., Borchert,T.V., Stemmer,W.P. and Minshull,J. (1999) *Nat. Biotechnol.*, **17**, 893–896.

Ness,J.E., Kim,S.,Gottman,A., Pak,R., Krebber,A., Borchert,T.V., Govindarajan,S., Mundorff,E.C. and Minshull,J. (2002) *Nat. Biotechnol.*, **20**, 1251–1255.

Nikolova,P.V., Henckel,J., Lane,D.P. and Fersht,A.R. (1998) *Proc. Natl Acad. Sci. USA*, **95**, 14675–14680.

Ostermeier,M., Shim,J.H. and Benkovic,S.J. (1999) *Nat. Biotechnol.*, **17**, 1205–1209.

Pierce,A.N. and Winfree,E. (2002) *Protein Eng.*, **15**, 779–782.

Sandberg,W.S. and Terwilliger,T.C. (1993) *Proc. Natl Acad. Sci. USA*, **90**, 8367–8371.

Schuster,P. and Stadler,P.F. (1994) *Comput. Chem.*, **18**, 295–324.

Sieber,V., Martinez,C.A. and Arnold,F.H. (2001) *Nat. Biotechnol.*, **19**, 456–460.

Smith,J.M. (1970) *Nature*, **225**, 563–564.

So,S.S. and Karplus,M. (1996) *J. Med. Chem.*, **39**, 1521–1530.

Spiller,B., Gershenson,A., Arnold,F.H. and Stevens,R.C. (1999) *Proc. Natl Acad. Sci. USA*, **96**, 12305–12310.

Stemmer,W.P.C. (1994) *Nature*, **370**, 389–391.

Taverna,D.M. and Goldstein,R.A. (2002) *J. Mol. Biol.*, **315**, 479–484.

Vajdos,F.F., Adams,C.W., Breece,T.N., Presta,L.G., de Vos,A.M. and Sidhu,S.S. (2002) *J. Mol. Biol.*, **320**, 415–428.

Weiss,G.A., Watanabe,C.K., Zhong,A., Goddard,A. and Sidhu,S.S. (2000) *Proc. Natl Acad. Sci. USA*, **97**, 8950–8954.

Wells,J.A. (1990) *Biochemistry*, **29**, 8509–8517.

Wold,H. (1981) *The Fix-Point Approach to Interdependent Systems*. North Holland, Amsterdam.

Wold,H. (1985) Partial least squares. In Kotz,S. and Johnson,N.L. (eds), *Encyclopedia of Statistical Sciences*, Vol. 6. Wiley, New York, NY, pp. 581–591.

Wright,S. (1932) *Proc. VI Intl. Congress Genet.*, **1**, 356–366.

Zhao,H., Giver,L., Shao,Z., Affholter,J.A. and Arnold,F.H. (1998) *Nat. Biotechnol.*, **16**, 258–261.

Received January 2, 2003; revised May 13, 2003; accepted June 19, 2003

Downloaded from http://peds.oxfordjournals.org/ at Aga Khan University on March 10, 2015