Nucleic Acids Research, 1998, Vol. 26, No. 3 **697–702**

Scoring functions for computational algorithms applicable to the design of spiked oligonucleotides

Lars Juhl Jensen, Kim Vilbour Andersen, Allan Svendsen and Titus Kretzschmar*

Department of Enzyme Design, Novo Nordisk A/S, DK-2880 Bagsværd, Denmark

Received December 3, 1997; Accepted December 4, 1997

ABSTRACT

Protein engineering by inserting stretches of random DNA sequences into target genes in combination with adequate screening or selection methods is a versatile technique to elucidate and improve protein functions. Established compounds for generating semi-random DNA sequences are spiked oligonucleotides which are synthesised by interspersing wild type (wt) nucleotides of the target sequence with certain amounts of other nucleotides. Directed spiking strategies reduce the complexity of a library to a manageable format compared with completely random libraries. Computational algorithms render feasible the calculation of appropriate nucleotide mixtures to encode specified amino acid subpopulations. The crucial element in the ranking of spiked codons generated during an iterative algorithm is the scoring function. In this report three scoring functions are analysed: the sum-of-square-differences function *s*, a modified cubic function *c*, and a scoring function *m* derived from maximum likelihood considerations. The impact of these scoring functions on calculated amino acid distributions is demonstrated by an example of mutagenising a domain surrounding the active site serine of subtilisin-like proteases. At default weight settings of one for each amino acid, the new scoring function *m* is superior to functions *s* and *c* in finding matches to a given amino acid population.

INTRODUCTION

With the advent of efficient site-directed mutagenesis methods, protein engineering became a widely-used method for analysing structure–function relationships (1,2). It has produced a wealth of information on protein functions and is a precious tool for the closer understanding of binding events, enzymatic activity, or stability of proteins on the molecular level. Site-directed mutagenesis is typically guided by some pre-knowledge of structural features of the protein and putative binding sites or conserved motifs.

In order to obtain higher degrees of diversity at a certain region of the protein and to accelerate the protein engineering process, random saturation mutagenesis (or combinatorial cassette mutagenesis) has been introduced (3–6). One or more codons consisting of equimolar mixtures of up to all four nucleotides at each position (NNN) are inserted into a gene. By this technique libraries of variants encoding all 20 natural amino acids plus the translation stop signals are created at a defined region of the gene which are then screened or selected by appropriate assays. This type of library encodes a biased pool of amino acids according to the degeneracy of the codons. A more parsimonious approach is using NN(G/C) or NN(G/T) codons which reduce the complexity at the DNA level by a factor of two per codon and result in a more even distribution of the amino acids with a smaller percentage of stop signals (7,8). NN(G/C) or NN(G/T) mutagenesis confines the window of the mutagenic area to about five codons, i.e., 3.4×10^7 variants at the DNA level with ~15% harbouring stop codons, because of limits in transforming host cells. A major disadvantage with these kinds of libraries, however, is the dilution of functionally active proteins by, e.g., misfolded, catalytically inactive, or prematurely truncated variants due to the very high density of mutant positions (9,10).

In order to overcome the constraints in window size and to drive the mutagenesis towards a larger fraction of functional molecules, three different strategies have evolved which aim at creating specified amino acid pools at certain positions based either on phylogenetic considerations, e.g., sequence alignments, on structurally derived properties of the protein, or simply on guesses.

Firstly, split-and-mix methods have been devised where during oligonucleotide synthesis the resin is split, differently processed, and mixed before proceeding with synthesis (11,12). The result is oligonucleotide pools exactly encoding a reference amino acid distribution. This approach becomes experimentally impractical when several complex split-and-mix steps are required.

Another elegant and efficient way of mutagenesis is the application of pre-formed trinucleotide phosporamidites for mixed oligodeoxyribonucleotide synthesis which also allows precise adjustment of a specified amino acid subset (13–17). The trinucleotide phosporamidites technique is not yet widespread and reagents are not commercially available.

Today's most applied method is exploiting spiked oligonucleotides. During oligonucleotide synthesis wild type (wt) bases are deliberately mixed with certain amounts of the other bases (18–25). Due to constraints imposed by the natural genetic code, spiking of oligonucleotides frequently does not result in perfect matches to a given amino acid distribution. Thus, the spiking approach has to consider the generation of an amino acid population which is as close to the reference amino acid set as possible while keeping the amount of stop codons low (12,26–28). In order to enable the calculation of appropriate

*To whom correspondence should be addressed at present address: MorphoSys GmbH, Am Klopferspitz 19, D-82152 Martinsried, Munchen, Germany. Tel: +49 89 89927 222; Fax: +49 89 899 222; Email: tik@morphosys.de

nucleotide mixtures, algorithms eligible to computer application have been developed. The essential element of these algorithms is the scoring function which allows for ranking of the spiked codons according to the best fit to a given amino acid population (12,26, this report).

We designed three scoring functions and discuss here their influence on semi-random mutagenesis with spiked oligonucleotides. We have chosen an example of mutagenesis of subtilisin-like proteases. Subtilases are of significant importance in detergent industry. Protein engineering, modelling and investigations on the mechanism of action as well as on the considerable stability of subtilisins have been performed for some time (29–33).

METHODS

Outline of the algorithm

The algorithm consists of 100 or more cycles and each cycle of 1000 iterations. Each iteration is based on a Monte-Carlo simulation: starting from any nucleotide composition of a codon a new random nucleotide composition is chosen for each position in the codon. The absolute difference between the starting and the new composition is controlled by a parameter *d* for each of the four nucleotides in all three positions of the codon. The parameter *d* is only kept constant within one iteration and decreases linearly from one to zero within one cycle. The encoded amino acid distribution is then scored by the respective scoring functions *s*, *c*, or *m* (see below) and compared with the starting score. If the new score is better or equal to the score of the current base composition, the new base composition will be the next starting point. If the new score is worse, the probability of keeping the new composition as new starting point equals $e^{(-1000 \times |new\_score - current\_score|)}$. The term 'better score' means, in the case of function *s* and function *c*, a lower score, and in the case of function *m* a higher score obtained by the algorithm. The entire algorithm was written in C++. It is compiled with the GNU C compiler and runs under Linux on a standard 100 MHz Pentium personal computer with 16 Mb of memory capacity. Performing 1000 cycles consisting each of 1000 iterations takes ~4 min of computing time.

Scoring functions

(i) The function *s* is derived from the least square sum method. A basically equivalent function is also used by Tomandl *et al.* (12):

$$s = \sum_{i=1}^{n} \{(a_i - x_i)^2 \times (w_i / \sum_{i=1}^{n} w_i)\}$$

with x_i defined as the fraction of an amino acid species *i* calculated by the algorithm for the current base composition, a_i is the reference fraction of amino acid species *i* as given by the user, and w_i is a user-specified weighting factor allowing for amino acids of special interest to be favoured. Default weight setting for all herein presented scoring functions is one.

(ii) The function *c* is based on a cubic sum calculation:

$$c = \sum_{i=1}^{n} \{\alpha_i \times |a_i - x_i|^3 \times (w_i / \sum_{i=1}^{n} w_i)\} \quad \text{with} \quad \begin{array}{ll} \alpha_i = a_i^{-3} & \text{if } x_i \le a_i, \\ \alpha_i = (1-a_i)^{-3} & \text{if } x_i > a_i, \end{array}$$

with the same definitions of x_i, a_i and w_i as described above.

(iii) The function *m* is derived from maximum-likelihood considerations:

$$m = \prod_{i=1}^{n} [\{x_i^{ai} \times (1-x_i)^{1-ai}\}^{wi} \times \{a_i^{ai} \times (1-a_i)^{1-ai}\}^{-wi}]$$

with the same definitions of x_i, a_i and w_i as described above. The term 0^0 is defined as being one.

Statistics on the mutagenised cassette

Average number of mutations. The average number of mutations per molecule is estimated from randomly generated variants by taking into account the probability of getting a non-wt amino acid at the individual positions. The number of mutations and of stop codons in each variant is counted and the average number of mutations is calculated on the basis of all species harbouring no stop codons. One thousand or more variants are generated until the estimated standard deviation of the average is <0.05.

Calculation of library sizes. For the calculation of the library size that has to be screened to cover at a 95% confidence level all demanded variants having *n* mutant sites, the probability of the least likely specified variant with *n* mutant sites has to be determined. The following algorithm is applied:

(i) The probability *p-wt* of getting wt amino acids at all positions is calculated.

(ii) For every position *i* the ratio r_i of the probability of getting the specified amino acid with the lowest amount divided by the probability of getting wt is calculated.

(iii) To find the probability p_1 of the least likely single mutant, the probability *p-wt* is multiplied with the smallest of the ratios r_i.

(iv) The probability p_2 of the least likely double mutant is found by multiplying p_1 with the second smallest ratio r_i.

(v) The probabilities p_3–p_n are analogously calculated.

(vi) The library size which has to be screened to cover all variants with *n* mutant sites at a 95% confidence level equals $\log(1 - 0.95)/\log(1 - p_n)$.

Subtilisin mutagenesis

The influence of the scoring functions *s*, *c* and *m* on the encoded amino acid population of mixed oligonucleotides and on the library size is shown here by the mutagenesis of six amino acids within a 10 amino acid stretch encompassing the active site serine of subtilisin proteases. For the analysis, 35 subtilisin sequences of class I type are used (30). We extracted the following distribution of amino acids at the positions 218–227 (Table 1). The wt amino acid is defined as the most abundant amino acid in the compilation of the position in question (e.g., serine in position 218, see Table 1).

Robustness of the calculated distribution of bases towards deviations caused by the imprecision of oligonucleotide synthesising machines

The calculated base composition in a codon encoding a given amino acid distribution is deliberately randomised imitating a 1%, 5% or 10% deviation range of the oligonucleotide synthesisers. For example, when assuming a maximum of a 10% error of the synthesiser, instead of 5% of A at a certain position in the codon, random numbers ranging from 0% to 15% of A were generated. The corresponding encoded amino acid distribution is then calculated. By repeating this algorithm 10^5 times a standard deviation for the amount of each amino acid as a function of the maximum error possible of the oligonucleotide synthesiser is obtained (Table 8).

Table 1. Distribution of amino acids in percentage at positions 218–227 of 35 compiled subtilisins class I (30)

	218	219	220	221	222	223	224	225	226	227
Ala	2.9				17.1	88.6	31.4			22.9
Asn	22.8									
Asp	2.9									
Cys							5.7			
Gln					2.9					
Glu									5.7	
Gly	8.6	100					2.9		2.9	
His									68.5	
Ile										5.7
Leu					5.7				17.1	
Lys									2.9	
Met					71.4					
Phe	2.9								2.9	
Pro								100		
Ser	48.5			100	2.9	11.4	20.0			
Thr	11.4		100				40.0			
Val										71.4

RESULTS AND DISCUSSION

In this study we present a new scoring function *m* which meets the requirements of both getting all requested amino acids and giving reasonable initial solutions even at default weight settings. This will keep the possible subsequent optimisation of spiked oligonucleotides by weight changes in the function *m* at a minimum.

Scoring functions

Before designing function *m* we looked at the function *s*, the sum-of-square-differences. In simple cases like aiming at 50% of cysteine and 50% of methionine residues problems emerged with function *s*. The anticipated solution is the codon (A/T)(G/T)(G/T) encoding 12.5% of each cysteine, methionine, arginine, isoleucine, leucine, phenylalanine, serine and tryptophan. With all weights at their default settings, this codon results in a score of 0.0179. But the triplet (T)(G)(25%G/25%A/50%T) yielding 50% cysteine, 25% tryptophan and 25% stops also gives a score of 0.0179 even though no methionine is obtained. In contrast, by exploiting the sum-of-cubic-differences function *c* for scoring, the expected (A/T)(G/T)(G/T) codon is always found. Another problem inherent to the function *s* is the loss of amino acids of which only small amounts were requested. We observed in several instances that one or more amino acids were lacking in the calculated solutions. An illustrative example is depicted in Table 6: neither glycine, leucine, nor lysine are obtained by the scoring function *s* at default settings (see Table 6, column s-def.). It should be noted that the problem could be overcome by optimising the weights, although in a time-consuming trial and error approach (see Table 6, column s-opt.). The underlying problems with scoring function *s* are its symmetry and the use of absolute instead of relative differences.

In order to circumvent these pitfalls the asymmetric, cubic scoring function *c* was designed. We noticed that with function *c* small amounts of specified, non-wt amino acids are easily preserved, however, in some instances unsatisfactory low amounts of the wt amino acid are retained (see for example Table 3, column c-def.) By increasing the weight for wt, it usually is possible but again rather time-consuming to come to a reasonable compromise between getting all the minor amino acids and an acceptable amount of wt. As functions *c* and *s* are sum functions, specified amino acids might be lacking in the calculated solutions, although such a case has not yet been observed with function *c*.

Table 2. Distributions of amino acids in percentage at position 218 of subtilisin class I (30), and the resulting amino acid distribution when either scoring function *s*, *c*, or *m* with default (def.) or optimised (opt.) weights are applied for the design of the respective spiked codons

	Pos.218	s-def.	s-opt.	c-def.	c-opt.	m-def.	m-opt.
Ala	2.9	1.7	0.9	2.6	2.1	1.9	1.9
Arg			0.8				
Asn	22.8	23.5	20.9	18.4	15.2	21.2	18.1
Asp	2.9	3.5	2.7	4.7	3.2	3.6	3.0
Cys		0.3	6.0	6.4	8.5	2.1	5.0
Gln							
Glu			0.1				
Gly	8.6	7.7	5.7	7.4	7.1	8.1	7.0
His							
Ile			7.0	10.4	9.5	2.2	6.4
Leu							
Lys			0.4				
Met			0.1				
Phe	2.9		1.0	2.3	2.4	0.1	0.8
Pro							
Ser	48.5	51.5	43.5	31.0	36.4	48.1	43.5
Thr	11.4	11.7	7.0	10.0	10.0	11.4	11.3
Trp			0.1				
Tyr		0.2	2.9	4.1	3.8	0.9	2.2
Val			0.6	2.7	2.0	0.4	1.1
Stop			0.05				

Table 3. Distributions of amino acids in percentage at position 222 of subtilisin class I (30), and the resulting amino acid distribution when either scoring function *s*, *c*, or *m* with default (def.) or optimised (opt.) weights are applied for the design of the respective spiked codons.

	Pos.222	s-def.	s-opt.	c-def.	c-opt.	m-def.	m-opt.
Ala	17.1	0.3	0.5	10.7	2.3	2.3	4.5
Arg		1.5	1.0				
Asn		0.1					
Asp							
Cys							
Gln	2.9	0.4	0.2	2.3	1.1	0.1	0.7
Glu		0.2	0.2	4.4	1.3	0.3	1.3
Gly		0.1	0.1				
His			0.1				
Ile		2.4					
Leu	5.7	8.6	8.9	11.6	13.5	8.1	9.2
Lys		4.1	2.0	6.7	7.4	1.8	4.4
Met	71.4	72.4	72.9	23.4	48.5	61.7	47.5
Phe		0.1					
Pro		0.5	0.5	5.7	1.9	0.9	2.3
Ser	2.9	0.2	0.2	2.5	1.7	0.8	0.7
Thr		5.4	5.9	16.5	12.7	13.0	15.0
Trp							
Tyr							
Val		3.6	6.2	15.1	8.8	10.9	14.3
Stop		0.11	0.07	1.02	0.96	0.12	0.19

Finally, a completely different approach was chosen for the design of the scoring function *m*. It was inspired by the likelihood function for the polynomial distribution. Function *m* is attaining values in the range [0;1]. The maximum value of one is only obtained for a perfect match with the target amino acid subset. This is also valid when aiming at non-identical amounts for the amino acid species in the given target pool. Because of the latter feature and the possibility for putting weight factors on each single amino acid, function *m* is much more generally applicable than the earlier described function P_G (26). Compared to the functions *s* and *c*, function *m* has two major advantages. Firstly, if any reference amino acid is missing, the score will be zero rejecting any incomplete solution. Secondly, the function *m* at default weight settings gives solutions where all specified minor

Table 4. Distributions of amino acids in percentage at position 223 of subtilisin class I (30), and the resulting amino acid distribution when either scoring function *s*, *c*, or *m* with default (def.) or optimised (opt.) weights are applied for the design of the respective spiked codons

	Pos.223	s-def.	s-opt.	c-def.	c-opt.	m-def.	m-opt.
Ala	88.6	88.5	88.5	88.6	88.6	88.6	88.6
Arg							
Asn							
Asp							
Cys							
Gln							
Glu							
Gly							
His							
Ile							
Leu							
Lys							
Met							
Phe							
Pro							
Ser	11.4	11.5	11.5	11.4	11.4	11.4	11.4
Thr							
Trp							
Tyr							
Val							
Stop							

Table 5. Distributions of amino acids in percentage at position 224 of subtilisin class I (30), and the resulting amino acid distribution when either scoring function *s*, *c*, or *m* with default (def.) or optimised (opt.) weights are applied for the design of the respective spiked codons

	Pos.224	s-def.	s-opt.	c-def.	c-opt.	m-def.	m-opt.
Ala	31.4	31.8	26.1	26.9	28.6	29.3	28.9
Arg							
Asn							
Asp							
Cys	5.7	1.5	3.5	4.9	3.9	2.7	3.3
Gln							
Glu							
Gly	2.9	3.5	4.5	8.7	7.3	4.5	5.5
His							
Ile							
Leu							
Lys							
Met							
Phe							
Pro							
Ser	20.0	20.0	26.8	26.1	24.5	23.5	24.3
Thr	40.0	37.7	39.2	32.4	35.6	40.1	38.1
Trp							
Tyr							
Val							
Stop							

components and the wt fraction are kept at a balanced level (e.g., compare columns s-def., c-def. and m-def. in Tables 3 and 6). In combination with the fact that function *m* is much more responsive to weight changes than functions *s* or *c* because the weights are entered as powers, the optimisation of function *m* for a user-specified purpose is significantly less time consuming (by a factor of 5–10).

Example of subtilisin mutagenesis

The performance of these three scoring functions is examined by mutagenising a domain of subtilisin-like proteases. The domain comprises 10 amino acid sites (positions 218–227 in subtilisin BPN′) of which four are strictly conserved and the residual six sites shall be mutagenised to match as precisely as possible the amino acid distribution derived from compilation of 35 subtilases.

In positions 223 (Table 4), 224 (Table 5), and 227 (Table 7) the reference amino acid distributions were closely matched by all three scoring functions without having to change weights. The corresponding codon composition is stable against errors which might occur during oligonucleotide synthesis (e.g., Table 8, position 224).

As for position 218 (Table 2) the scoring functions were providing reasonable results. With function *s* at default weight settings the requested phenylalanine was not present in the solution. The function *m* with default weights also yielded very small amounts of phenylalanine. By modifying the weights, the percentages of phenylalanine could be increased to acceptable values. As expected, the function *c* yielded relatively high amounts of the specified minor components alanine, aspartate and phenylalanine, at the expense of the major components serine and asparagine.

Regarding position 222 (Table 3), a matching combination for the five specified amino acids is difficult to obtain as can easily be deduced from codon tables. All found solutions generated at least 10 different amino acids. It is mutually excluding to get ~17% alanine and >70% methionine. The function *s* had no difficulty in identifying spiked codons encoding ~70% methionine but at the cost of alanine, while the function *c* had tremendous problems in preserving methionine. The function *m* at default weight settings resulted in a compromise between both extremes. Again, the solution calculated with function *m* is robust towards potential, non-deliberately introduced deviations by oligonucleotide synthesisers (Table 8, position 222). We noticed that it was impossible within a reasonable time-frame to optimise function *s* by trial and error weight adjustments without losing reference amino acids.

In position 226 (Table 6) it is impossible to obtain a high quality solution due to constraints imposed by the genetic code. With function *s* at default weight settings, glycine, leucine and lysine were missed. It was not trivial to find a set of weights driving the amino acid pool to contain all specified amino acids. In the course

Table 6. Distributions of amino acids in percentage at position 226 of subtilisin class I (30), and the resulting amino acid distribution when either scoring function *s*, *c*, or *m* with default (def.) or optimised (opt.) weights are applied for the design of the respective spiked codons

	Pos.226	s-def.	s-opt.	c-def.	c-opt.	m-def.	m-opt.
Ala							
Arg			3.2	5.5	5.2	1.8	3.4
Asn			6.2	5.6	4.0	2.0	3.0
Asp		1.6	11.1	10.6	9.1	5.0	8.3
Cys			0.4	1.0	0.4		0.1
Gln		2.3	11.1	9.7	8.1	7.0	8.7
Glu	5.7	0.1	2.9	4.1	1.4	0.6	1.3
Gly	2.9		0.8	2.2	0.9	0.1	0.5
His	68.5	70.1	42.1	25.0	54.1	64.2	54.5
Ile			0.9	2.3	0.5	0.4	0.7
Leu	17.1		7.6	13.3	8.1	15.2	13.2
Lys	2.9		1.6	2.1	0.6	0.2	0.5
Met			0.2	0.5	0.1		0.1
Phe	2.9	0.7	0.9	2.4	0.6	0.3	0.5
Pro		1.1					
Ser			0.4	0.8	0.3		0.2
Thr							
Trp			0.1	0.2			
Tyr		2.3	6.7	6.7	4.7	1.6	2.6
Val		0.5	2.0	5.3	1.3	1.2	2.0
Stop		0.08	1.78	2.75	0.71	0.17	0.42

Table 7. Distributions of amino acids in percentage at position 227 of subtilisin class I (30), and the resulting amino acid distribution when either scoring function *s*, *c*, or *m* with default (def.) or optimised (opt.) weights are applied for the design of the respective spiked codons

	Pos.227	s-def.	s-opt.	c-def.	c-opt.	m-def.	m-opt.
Ala	22.9	21.7	21.7	22.2	22.2	22.1	22.1
Arg							
Asn							
Asp							
Cys							
Gln							
Glu							
Gly							
His							
Ile	5.7	4.8	4.8	5.7	5.7	4.3	4.3
Leu							
Lys							
Met				0.2	0.2		
Phe							
Pro							
Ser							
Thr		1.4	1.4	1.9	1.9	1.3	1.3
Trp							
Tyr							
Val	71.4	72.1	72.1	70.0	70.0	72.3	72.3
Stop							

Table 8. Robustness of the amino acid distributions towards deviations in the base composition of codons caused by the imprecision of the oligonucleotide synthesising machines

	Pos.222 m-def.	1% error	5% error	10% error	Pos.224 m-def.	1% error	5% error	10% error
Ala	2.3	2.3±0.11	2.2±0.55	2.1±1.08	29.3	29.1±0.48	28.3±2.27	27.3±4.39
Arg		0.2±0.26	1.0±1.24	1.8±2.35		0.1±0.05	0.3±0.25	0.6±0.52
Asn		0.0±0.01	0.1±0.07	0.1±0.21		0.1±0.15	0.5±0.69	1.0±1.29
Asp		0.0±0.00	0.0±0.01	0.0±0.04		0.1±0.11	0.4±0.50	0.7±0.95
Cys		0.0±0.00	0.0±0.00	0.0±0.01	2.7	2.6±0.12	2.5±0.58	2.3±1.11
Gln	0.1	0.1±0.03	0.1±0.15	0.2±0.30		0.0±0.00	0.0±0.00	0.0±0.01
Glu	0.3	0.3±0.08	0.4±0.32	0.5±0.54		0.0±0.00	0.0±0.02	0.0±0.07
Gly		0.0±0.04	0.2±0.22	0.3±0.44	4.5	4.5±0.18	4.3±0.89	4.1±1.75
His		0.0±0.00	0.0±0.01	0.0±0.02		0.0±0.00	0.0±0.03	0.1±0.13
Ile		0.5±0.34	2.2±1.57	4.0±2.90		0.1±0.15	0.5±0.70	1.0±1.33
Leu	8.1	8.0±0.60	7.7±2.94	8.2±4.83		0.0±0.00	0.0±0.04	0.1±0.15
Lys	1.8	1.8±0.43	2.0±1.76	2.6±2.77		0.0±0.00	0.0±0.03	0.1±0.09
Met	61.7	61.1±0.90	58.8±4.15	55.3±7.27		0.0±0.00	0.0±0.02	0.0±0.06
Phe		0.0±0.02	0.1±0.11	0.2±0.27		0.1±0.06	0.2±0.30	0.4±0.58
Pro	0.9	0.9±0.10	0.9±0.48	0.9±0.83		0.2±0.28	1.0±1.32	2.0±2.54
Ser	0.8	0.8±0.10	0.8±0.48	0.9±0.83	23.5	23.3±0.44	22.5±2.11	21.5±4.10
Thr	13.0	12.9±0.40	12.8±2.00	12.3±3.90	40.1	39.8±0.54	38.7±2.56	37.5±4.94
Trp		0.0±0.02	0.1±0.09	0.1±0.22		0.0±0.01	0.0±0.04	0.1±0.08
Tyr		0.0±0.00	0.0±0.01	0.0±0.02		0.1±0.06	0.2±0.30	0.4±0.57
Val	10.9	10.9±0.43	10.7±2.12	10.3±4.11		0.1±0.11	0.4±0.52	0.8±0.99
Stop	0.12	0.1±0.03	0.1±0.15	0.2±0.30		0.0±0.01	0.0±0.05	0.1±0.10

Assuming perfectly synthesised spiked oligonucleotides, the encoded amino acid distribution of the codons obtained with function *m* at default weight settings for subtilisins class I positions 222 and 224 are given (pos.222 m-def., pos.224 m-def., see also Tables 3 and 5). Assuming deviations of 1%, 5% or 10% during the oligonucleotide synthesising process, the respective changes in the amino acid distribution are calculated as described in Methods (mean ± standard deviation).

of optimising the weights for all three scoring functions, function *m* proved to be superior with regard to time considerations.

Statistics on the numbers of mutations and library sizes

If one had exploited one of the techniques for generating perfect matches to the given amino acid distributions in the above

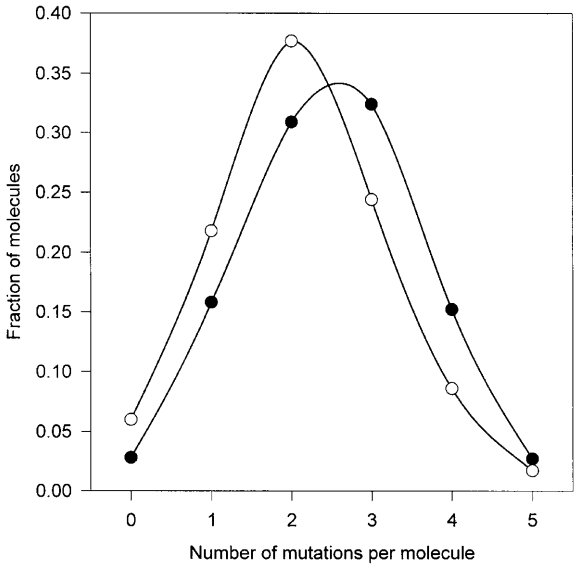

Figure 1. Distribution of mutant sites per molecule obtained when applying the scoring function *m* for the design of spiked codons. The distribution of mutant sites per molecule is calculated as described in Methods. (○) The ideal mutant distribution obtained by a spiked 30mer oligonucleotide which encodes the perfect match to all 10 reference amino acid populations as extracted from positions 218–227 of 35 compiled subtilisins (see Table 1). (●) The corresponding mutant distribution when the scoring function *m* with optimised weights is applied for the oligonucleotide design.

subtilisin example, the average number of mutations per molecule would have been 2.1. In contrast, applying the scoring function *m* at optimised weight settings gives ~2.5 mutant sites per molecule. The distributions of mutant sites per molecule are shown to be rather similar (Fig. 1). When aiming at covering with a 95% confidence level all transformants harbouring specified double or triple mutants, and when the function *m* is used for ranking the spiked codons, the corresponding libraries should consist of 8.0×10^5 and 4.4×10^7 members. These libraries are clearly larger than those obtained with an optimal technique where only 2.9×10^4 and 4.9×10^5 transformants have to be screened to cover all double and triple mutants, respectively. As many as 63% of all the members of the library created by the exploitation of the function *m* harbour at least one non-wanted mutation.

CONCLUSIONS

(i) If the reference amino acid distributions are readily accessible because of the absence of genetically impossible combinations, scoring functions *s*, *c* and *m* are working equally well.

(ii) The use of scoring function *s* may lead to a complete loss of minor components in the requested amino acid set.

(iii) The scoring function *c* strongly favours minor components in the specified amino acid sub-population at the expense of the major components.

(iv) The scoring function *m* generally yields good compromises between the functions *s* and *c*. It is unambiguously preferred because of the ease of handling the weight settings. Hence, it is the function of choice for designing spiked oligonucleotides.

REFERENCES

1 Smith,M. (1985) *Annu. Rev. Genet.*, **19**, 423–462.
2 Kunkel,T.A., Bebenek,K. and McClary,J. (1991) *Methods Enzymol.*, **204**, 125–139.
3 Hui,A., Hayflick,J., Dinkelspiel,K. and de Boer,H.A. (1984) *EMBO J.*, **3**, 623–629.
4 Botstein,D. and Shortle,D. (1985) *Science*, **229**, 1193–1201.
5 Wells,J.A., Vasser,M. and Powers,D.B. (1985) *Gene*, **34**, 315–323.
6 Oliphant,A.R., Nussbaum,A.L. and Struhl,K. (1986) *Gene*, **44**, 177–183.
7 Reidhaar-Olson,J.F. and Sauer,R.T. (1988) *Science*, **241**, 53–57.
8 Little,J.W. (1990) *Gene*, **88**, 113–115.
9 Delagrave,S. and Youvan,D.C. (1993) *Bio/Technology*, **11**, 1548–1552.
10 Suzuki,M., Christians,F.C., Kim,B., Skandalis,A., Black,M.E. and Loeb,L.A. (1996) *Mol. Diversity*, **2**, 111–118.
11 Glaser,S.M., Yelton,D.E. and Huse,W.D. (1992) *J. Immunol.*, **149**, 3903–3913.
12 Tomandl,D., Schober,A. and Schwienhorst,A. (1997) *J. Comp-Aided Mol. Design*, **11**, 29–38.
13 Chattopadhyaya,J.B. and Reese,C.B. (1980) *Nucleic Acids Res.*, **8**, 2039–2053.
14 Sondek,J. and Shortle,D. (1992) *Proc. Natl. Acad. Sci. USA*, **89**, 3581–3585.
15 Virnekäs,B., Ge,L., Plückthun,A., Schneider,K.C., Wellnhofer,G. and Moroney,S.E. (1994) *Nucleic Acids Res.*, **22**, 5600–5607.
16 Lyttle,M.H., Napolitano,E.W., Calio,B.L. and Kauvar,L.M. (1995) *BioTechniques*, **19**, 274–280.
17 Ono,A., Matsuda,A., Zhao,J. and Santi,D.V. (1995) *Nucleic Acids Res.*, **23**, 4677–4682.
18 Matteucci,M.D. and Heyneker,H.L. (1983) *Nucleic Acids Res.*, **11**, 3113–3121.
19 Dreher,T.W., Bujarski,J.J. and Hall,T.C. (1984) *Nature*, **311**, 171–175.
20 McNeil,J.B. and Smith,M. (1985) *Mol. Cell. Biol.*, **5**, 3545–3551.
21 Hutchison,C.A.,III, Nordeen,S.K., Vogt,K. and Edgell,M.H. (1986) *Proc. Natl. Acad. Sci USA*, **83**, 710–714.
22 Derbyshire,K.M., Salvo,J.J. and Grindley,N.D.F. (1986) *Gene*, **46**, 145–152.
23 Hermes,J.D., Parekh,S.M., Blacklow,S.C., Köster,H. and Knowles,J.R. (1989) *Gene*, **84**, 143–151.
24 Siderovski,D.P. and Mak,T.W. (1993) *Comput. Biol. Med.*, **23**, 463–474.
25 Ophir,R. and Gershoni,J.M. (1995) *Protein Engng*, **8**, 143–146.
26 Arkin,A.P. and Youvan,D.C. (1992) *Bio/Technology*, **10**, 297–300.
27 Balint,R.F. and Larrick,J.W. (1993) *Gene*, **137**, 109–118.
28 Schier,R., Balint,R.F., McCall,A., Apell,G., Larrick,J.W. and Marks,J.D. (1996) *Gene*, **169**, 147–155.
29 Wells,J.A. and Estell,D.A. (1988) *Trends Biochem. Sci.*, **13**, 291–297.
30 Siezen,R.J., de Vos,W.M., Leunissen,J.A.M. and Dijkstra,B.W. (1991) *Protein Engng*, **4**, 719–737.
31 Jarnagin,A.S. and Ferrari,E. (1992) *Biotechnology*, **22**, 189–217.
32 Bryan,P.N. (1995) In Shirley,B.A. (ed.), *Methods in Molecular Biology*. Humana Press Inc., Totowa, NJ, Vol. 40, pp. 271–289.
33 Siezen,R.J. and Leunissen,J.A.M. (1997) *Protein Sci.*, **6**, 501–523.