Protein Science (1999), *8*:680–688. Cambridge University Press. Printed in the USA.
Copyright © 1999 The Protein Society

FOR THE RECORD

Combinatorial codons: A computer program to approximate amino acid probabilities with biased nucleotide usage

ETHAN WOLF AND PETER S. KIM

Howard Hughes Medical Institute, Whitehead Institute, Department of Biology, MIT, Nine Cambridge Center, Cambridge, Massachusetts 02142

(RECEIVED July 29, 1998; ACCEPTED November 6, 1998)

Abstract: Using techniques from optimization theory, we have developed a computer program that approximates a desired probability distribution for amino acids by imposing a probability distribution on the four nucleotides in each of the three codon positions. These base probabilities allow for the generation of biased codons for use in mutational studies and in the design of biologically encoded libraries. The dependencies between codons in the genetic code often makes the exact generation of the desired probability distribution for amino acids impossible. Compromises are often necessary. The program, therefore, not only solves for the "optimal" approximation to the desired distribution (where the definition of "optimal" is influenced by several types of parameters entered by the user), but also solves for a number of "sub-optimal" solutions that are classified into families of similar solutions. A representative of each family is presented to the program user, who can then choose the type of approximation that is best for the intended application. The *Combinatorial Codons* program is available for use over the web from http://www.wi.mit.edu/kim/computing.html.

Keywords: codon bias; combinatorial libraries; computational techniques

Combinatorial and mutational studies of protein structures and functions often require an investigator to generate a nucleotide sequence encoding an amino acid sequence that has a bias toward specific properties. West and Hecht studied correlations between protein structures and hydrophobic patterning in amino acid sequences created with a simplified alphabet of two codon types: NAN (polar) and NTN (nonpolar) (West & Hecht, 1995). Other ad hoc methods for generating peptides with specific properties through biased codon usage have been used in the construction of combinatorial libraries and to perform mutational studies (Reidhaar-Olson & Sauer, 1988; Hu et al., 1993; Pu & Struhl, 1993). LaBean and Kauffman developed a spreadsheet-based method to partially automate the generation of bias in amino acids. Their method searches the space of potential distributions either by manually changing the nucleotide mixture, or by exhaustively searching for the best mixture in a specified subregion of the space of all mixtures (LaBean & Kauffman, 1993).

This paper describes a fully automated computer program designed to generate a probability distribution on amino acids by biasing codon usage. The program imposes a probability distribution on the four nucleotides in each of the three codon positions to either approximate a desired probability distribution defined for all amino acids, or to approximate a probability distribution defined for classes of amino acids. Such classes include size, hydrophobicity, charge, or other properties that the program user might care to define. The probability within each of these classes of amino acids can be arbitrarily distributed by the program among the individual amino acids in the class or can be skewed by additional user-imposed constraints. The base probabilities output by the program allow for the generation of the biased codons by creating a weighted mixture of the four nucleotides at each codon position.

Methods: *Approach:* The problem of approximating the desired distribution of probabilities for amino acids was formulated as an optimization problem. Each *potential* solution (corresponding to a particular probability distribution for the bases) was assigned an energy based upon the *differences* between the desired amino acid probabilities input by the user and the amino acid probabilities generated by the program. This energy function was used to evaluate the quality of potential solutions.

Computing the probability distribution on amino acids from the distribution on nucleotides: Let P_1, P_2, and P_3 denote the three probability functions defined by the program on the four nucleotides in the first, second, and third codon positions, respectively. The probability for codon $N_1N_2N_3$ is $P_1(N_1) \cdot P_2(N_2) \cdot P_3(N_3)$. Com-

Reprint requests to: Peter S. Kim, Howard Hughes Medical Institute, Whitehead Institute, Department of Biology, MIT, Nine Cambridge Center, Cambridge, Massachusetts 02142; e-mail: tocio@wi.mit.edu.

bining these values over different codons for an amino acid *aa* leads to a probability estimate (P_{est}) on the amino acid of

$$P_{est}(aa) = \sum_{C=N_1 N_2 N_3} P_1(N_1)\cdot P_2(N_2)\cdot P_3(N_3), \tag{1}$$

where the sum is over all codons *C* coding for *aa*. The probability of a class of amino acids is the sum of the probabilities of the amino acids in the class.

Computing the energy of an amino acid probability distribution: To find the "best" solution, an energy function measuring the quality of the solutions is required. Each amino acid or class of amino acids contributes to the energy based on the difference between the estimated probability distribution (P_{est}) output by the program and the desired probability distribution (P_{des}) input by the program user. The energy contribution from either a class of amino acids or from a single amino acid *c* is

$$E(c) = 1 - \cos(|P_{des}(c) - P_{est}(c)|\,\pi). \tag{2}$$

This energy function has a basin-like shape (Fig. 1) that changes slowly near the desired probability, increases, and then tapers off as the difference becomes large. In the calculation of the total energy (which involves combining many of these individual energy contributions), this shape allows a moderate difference between the desired and estimated probability for one amino acid or class to have a greater effect than the combined contribution from very small differences for a number of amino acids that are not considered as detrimental to the quality of the solution.

The total energy of an estimated probability distribution is computed as a weighted sum of the individual energy contributions for

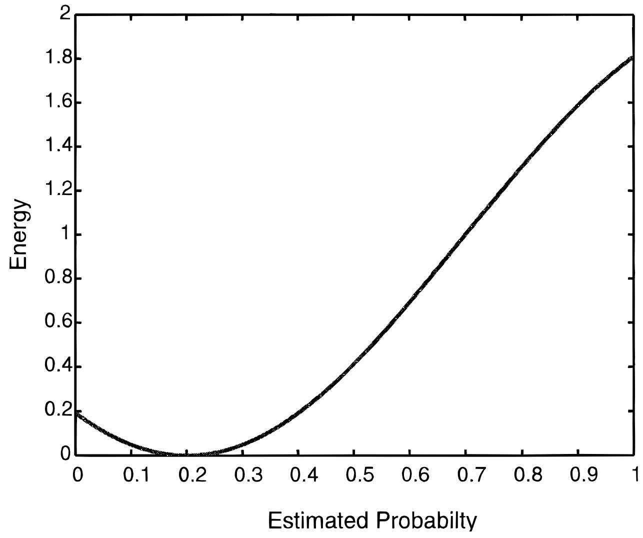

Fig. 1. The quality of an estimated probability distribution (P_{est}) for the amino acids is determined by energy contributions from each amino acid. The energy is based upon the difference between the estimated (P_{est}) and desired probabilities (P_{des}). The shape of the energy contribution function is plotted as P_{est} varies on the *x*-axis for $P_{des} = 0.2$.

c ranging over amino acids, stop codons, and classes of amino acids chosen by the program user:

$$E = \sum_c wt(c) * E(c). \tag{3}$$

The program user enters the weights $wt(c)$ to capture the relative importance of correctly estimating the desired probability for the *c*th class or amino acid. In addition, this weight can be set separately for overestimating the probability and for underestimating it (as it may be more detrimental to miss the probability in one direction than the other).

Solving for the "best" solution: The problem of finding a distribution on the four bases to approximate the desired probability distribution on amino acids can now be formulated as a problem in nonlinear optimization. The energy function in Equation 3 can be computed from Equations 1 and 2 as a nonlinear combination of the base probabilities $P_i(N)$. Finding the "best" solution is then a matter of minimizing this energy subject to the following linear constraints:

- Each of the four base probabilities $P_i(N)$ in the three codon positions $i = 1,2,3$ lies between 0 and 1:

$$0 \le P_i(N) \le 1 \quad \text{for } N = A,C,G,T \text{ and } i = 1,2,3.$$

- The probabilities sum to 1 for each codon position:

$$\sum_{N=A,C,G,T} P_i(N) = 1 \quad \text{for } i = 1,2,3.$$

Additional nonlinear constraints can also be specified to bound the computed probability of an amino acid (or stop codon or class) above or below particular values:

- $P_{est}(c) <$ upper bound and $P_{est}(c) >$ lower bound.

Similarly, the probability of generating a particular codon can also have an upper bound enforced. This feature can be useful in practice to steer the program solution away from rare codons, which may cause problems with protein expression (Kane, 1995).

Solving for locally minimal solutions: Often the probability distribution on amino acids desired by the user cannot be approximated exactly by a probability distribution on bases. For example, no base probability distribution can generate amino acids Tyr (codons TAT and TAC) and Trp (codon TGG) with nonzero probability, while also giving zero probability to stop codons (TAA, TAG, and TGA). There are several approximate solutions and the choice of the "best" solution depends on the intended application for the distribution. Thus, it is desirable to present the program user with a number of different types of solutions from which to choose.

To this end, the base probability space was divided into 125 subspaces, wherein each subspace was specified by placing bounds on the base probabilities in the three codon positions. For each codon position *i*, either one base N_i was bounded above 0.5 probability ($P_i(N_i) > 0.5$) while the other three base probabilities were bounded below 0.5, or all four bases were bounded below 0.5. These five different types of bounds result in $5^3 = 125$ regions of the entire probability space over all three codon positions. Minimizing the energy on each subspace generates 125 *potential* solutions.

A number of the solutions obtained may be unsatisfactory due to the lack of a good solution within a particular subspace. The program eliminates these bad solutions by discarding a fraction *f* with the largest energy. For the examples presented in this paper, 3/5 of the solutions were discarded.

In addition, many of the remaining solutions are redundant, generating the same or very similar probability distributions of amino acids. The solutions were therefore classified into *n* families of similar solutions, where *n* is determined by the program user. For the examples in the results section, five families were considered. Each family was represented by the member of the family with the smallest energy (corresponding to the best solution). To determine the families, a measure was defined to quantify the similarity between two probability distributions on amino acids. The distance between two such potential solutions P_{sol1} and P_{sol2} was defined to be the sum over all amino acids (including stop codons, but excluding classes) of the absolute value of the difference between the two probability values:

$$d(P_{sol1}, P_{sol2}) = \sum_{aa} |P_{sol1}(aa) - P_{sol2}(aa)|.$$

Solutions with similar distributions according to this distance measure were classified into the same family. A distance d_n was chosen (via a binary search of the possible distances *d* between pairs of solutions) to divide the solution space into the desired number of families *n*. The following procedure incrementally creates families of solutions by examining each solution in ascending order of energy (thereby looking at the best solutions first) and adding the solution to a currently existing family if it lies within distance *d* of the lowest energy member of the family, and otherwise making the solution the representative member of a *new* family.

- Compute the distances between all pairs of solutions. Order these distances in ascending order.
- Start with *d* equal to the median distance in the ascending list of all distances.
- Step through the solutions *S* in ascending order of energy.
- Find the representative solution *R* from the currently created list of families such that the distance between *S* and *R* is less than *d*. Place *S* in *R*'s family. If there are no such representatives, then make *S* the representative of a new family.
- If at any point the number of solutions in the current list of representatives is greater than *n*, then start over with *d* chosen to be the median in the upper half of the remaining possible distances.
- If all solutions are classified by the distance into fewer than *n* families, then start over with *d* chosen to be the median in the lower half of the remaining possible distances.

Solving the nonlinear program: The CFSQP optimization routines available at

http://www.isr.umd.edu/Labs/CACSE/FSQP/fsqp.html

were used to solve the nonlinear programs (Lawrence et al., 1997). A default probability of 0 and weight or relative importance of 1 was given to amino acids not explicitly specified by the program user. For the examples in Results, the amino acids in the classes and the entire class were given weight 10 with lower and upper bounds on the probabilities of 0 and 1, respectively. Stop codons were given weight 1, and the probability of generating a stop codon was additionally constrained to lie between 0 and 0.1 by imposing bounds.

Results: *Defining the test problems for the program:* To demonstrate the applicability and limitations of the program, several natural classes of amino acids were defined based on common amino acid properties. For purposes of illustration, we present results from using the program to generate codon distributions based upon some of the groupings of amino acids presented by Taylor (1986). The input parameters for the program were set to distribute the probability approximately equally among the individual amino acids within each class. The results from running the program to generate these distributions are available via the web at:

http://www.wi.mit.edu/kim/computing.html.

The website also allows for program users to generate their own distributions and define their own classes of amino acids to suit their experimental needs.

The properties used as examples and available at the website are:

1. Size:
   - Tiny: Cys,[1] Ser, Ala, Gly (Fig. 2)
   - Small: Asn, Thr, Asp, Pro, Cys,[1] Ser, Ala, Gly
   - Medium: Met, Leu, Ile, His, Gln, Glu, Val
   - Large: Trp, Arg, Tyr, Phe, Lys

   The relative size classifications were based upon the van der Waals volume of the amino acids (Richards, 1974).

2. Hydrophobic residues: Phe, Ile, Val, Leu, Met, Trp, His, Tyr, (Ala, Gly). A hydrophobicity scale (Rose et al., 1985) was used to determine the hydrophobic residues, using both Gly and Tyr as cutoff amino acids for two different trials. Despite its hydrophobic nature, cysteine was not included due to its ability to form disulfide bonds.

3. Polar: Arg, Lys, His, Glu, Asp, Gln, Asn, Thr, Ser (Figs. 3, 4).

4. Positive: Arg, Lys, (His). Histidine was considered in one trial run as positively charged and in another as noncharged since the pK_a of the histidine side chain is near seven.

5. Negative: Glu, Asp.

6. Charged: Arg, Lys, (His), Glu, Asp (Figs. 5, 6). For the desired probability in this example, half of the probability was distributed equally among the positively charged residues (Arg, Lys, and His), while the other half was distributed among the negative residues (Glu and Asp). Again, histidine was considered in one trial run as positively charged and in another as noncharged.

7. Aliphatic: Leu, Ile, Val.

8. Aromatic: Phe, Tyr, Trp.

9. Beta branched: Val, Ile, Thr.

[1]Program results for “tiny” and “small” with and without cysteine were computed, since experimental applications may often want to avoid cysteine residues due to potential disulfide bond formation.

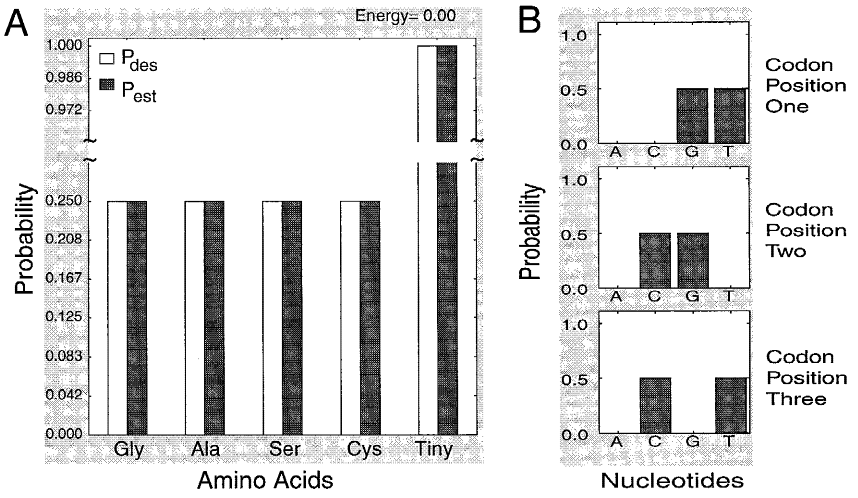

Fig. 2. Probability distributions for the class of tiny amino acids. **A:** Comparison between the distribution of probabilities P_{des} to be approximated (open bars) and the probabilities P_{est} generated by the program (filled bars). The *y*-axis gives the probability for each amino acid listed. The energy of the solution is given at the top in arbitrary units. **B:** The nucleotide probabilities generated by the program in each of the three codon positions to generate P_{est}. From top to bottom the panels plot the probabilities for the first, second, and third positions of the codon.

Note that when a stop codon received nonzero probability, the rest of the probabilities were normalized by $1 - P(STOP)$. The probability given in the examples for amino acid *aa* therefore represents the fraction of all generated nonstop codons coding for *aa*.

Analyzing the output for the test examples: For purposes of illustration we present representatives of the types of solutions output by the program for several of the classes defined in the previous section. The optimal solution to the nonlinear program for the class of tiny amino acids (including cysteine) yields an exact solution. Figure 2A shows a comparison between the desired probabilities input to the program as open bars and the generated probabilities output by the program as filled bars. The stacked charts in Figure 2B show the nucleotide probability values that generate the solution.

For other classes of amino acids, there was often no single solution that matched all of the desired probabilities. Additional amino acids that are outside the desired class are often generated with nonzero probability due to unavoidable dependencies within the genetic code. In these cases, the program was used to generate multiple potential solutions, each with different advantages and disadvantages in comparison to the ideal solution. These multiple solutions were created by running the program in multiple solution mode, which solves for the best solution on each of 125 subspaces of the entire space of nucleotide distributions. The program ranks these solutions based on their quality, and keeps the best 2/5 fraction of solutions that are then grouped into five families of similar solutions based upon the distance measure and the algorithm defined in the approach section. For the examples presented, we only show the top four families in each figure, since the last family is often of low quality (though not always). All five families are available for viewing at the website.

In the figures, the nucleotide variation among the solutions within each family is plotted as open bars (representing a range of probabilities) in the three stacked charts. This range shows the maximum and minimum probability values taken by each base within that family of solutions. The variability of this range gives a rough indication as to the accuracy required when experimentally generating the base probabilities. The range of tolerable accuracy can prove useful to the experimentalist generating the nucleotide distribution, as experimental systems have limitations on their accuracy. For more information on the accuracy required, it is important to consider the covariance between different nucleotide probabilities by examining the individual solutions within each family. For example, T and C nucleotides in the last position of a codon are interchangeable in the genetic code, which means that, within a family, covarying the probabilities of T and C while keeping their sum constant results in identical probability distributions on amino acids. Since the program only identifies one minimal energy solution on each of the 125 subspaces, any variability within base probabilities in a single subspace is not accessible to the program user. For instance, the second solution in Figure 5 allows for more variability among the T and C nucleotides than shown, as long as the total probability allotted to T and C remains constant. The accuracy required to generate the biased codons should be evaluated on a case by case basis by the program user.

Figure 3 shows an example of the top four families generated for the class of polar amino acids. While none of the solutions can exactly match the desired amino acid probabilities, the solutions each distribute the probability with different biases, giving the program user the choice of which solution to use. The comparison plot of the amino acid distributions for each family shows the differences between the desired amino acid probabilities (open bars) and the generated amino acid probabilities (filled bars) for the lowest energy solution within the family. The energy of this solution (in arbitrary units) is indicated in each case. The nucleotide probabilities that generate this lowest energy solution are plotted as narrow filled horizontal bars in the three stacked charts (corresponding to the nucleotide probabilities for the first, second, and third codon positions). The open horizontal bars for the nucleotide probabilities show the maximum and minimum values taken by that base within the family.

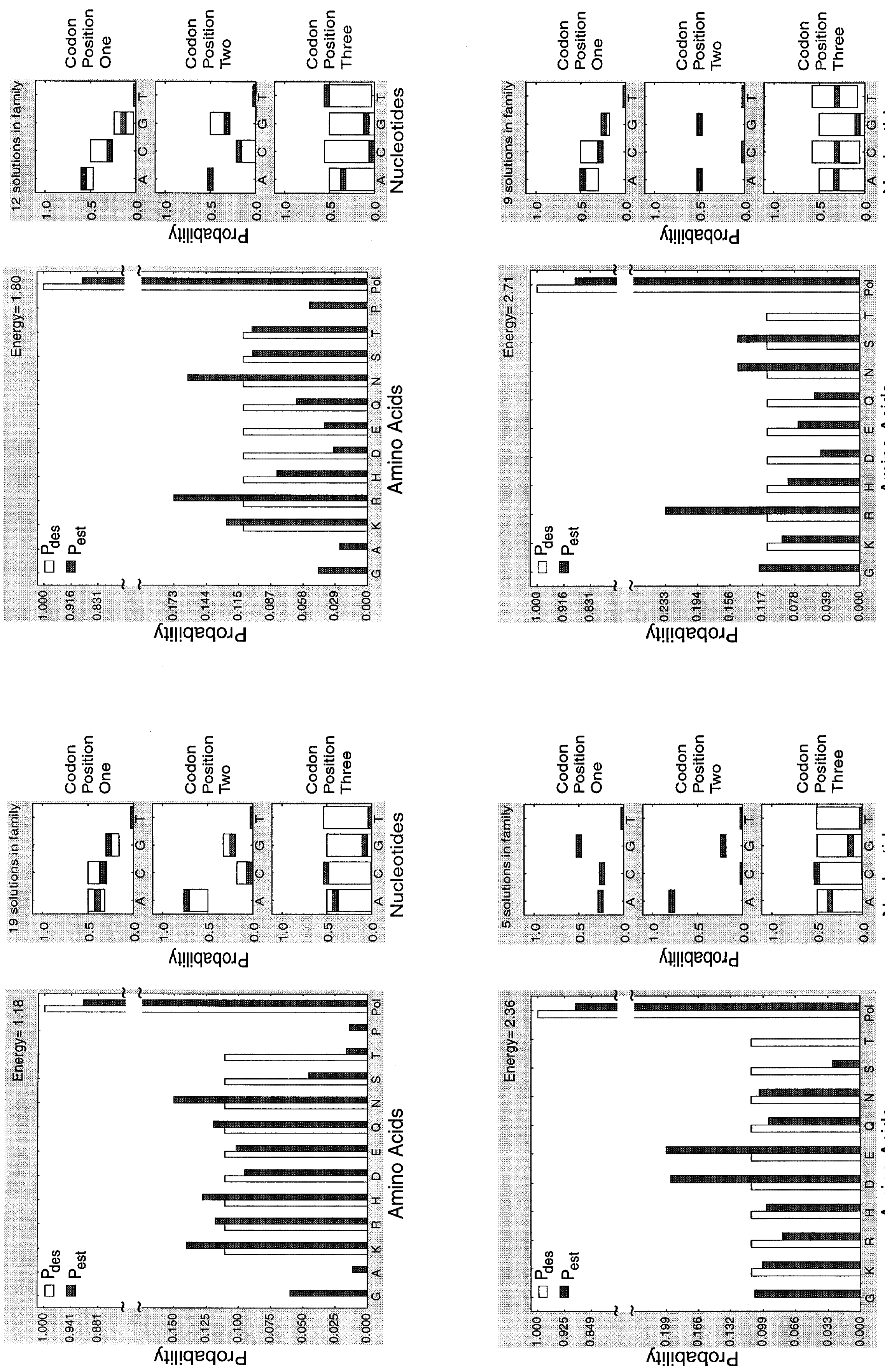

Fig. 3. Plots of the four families of solutions with lowest energy for the class of polar amino acids. The **left panel of each pair** compares the desired probability distribution on amino acids (open bars) with the generated solution (filled bars). The **right panel of each pair** plots the range of probabilities taken by each nucleotide in the three codon positions (open horizontal bars) for the solutions in the family. The filled horizontal bars for each base position show the probabilities that generate the representative solution plotted in the left panel of the pair. The number of solutions in the family is also given along the top.

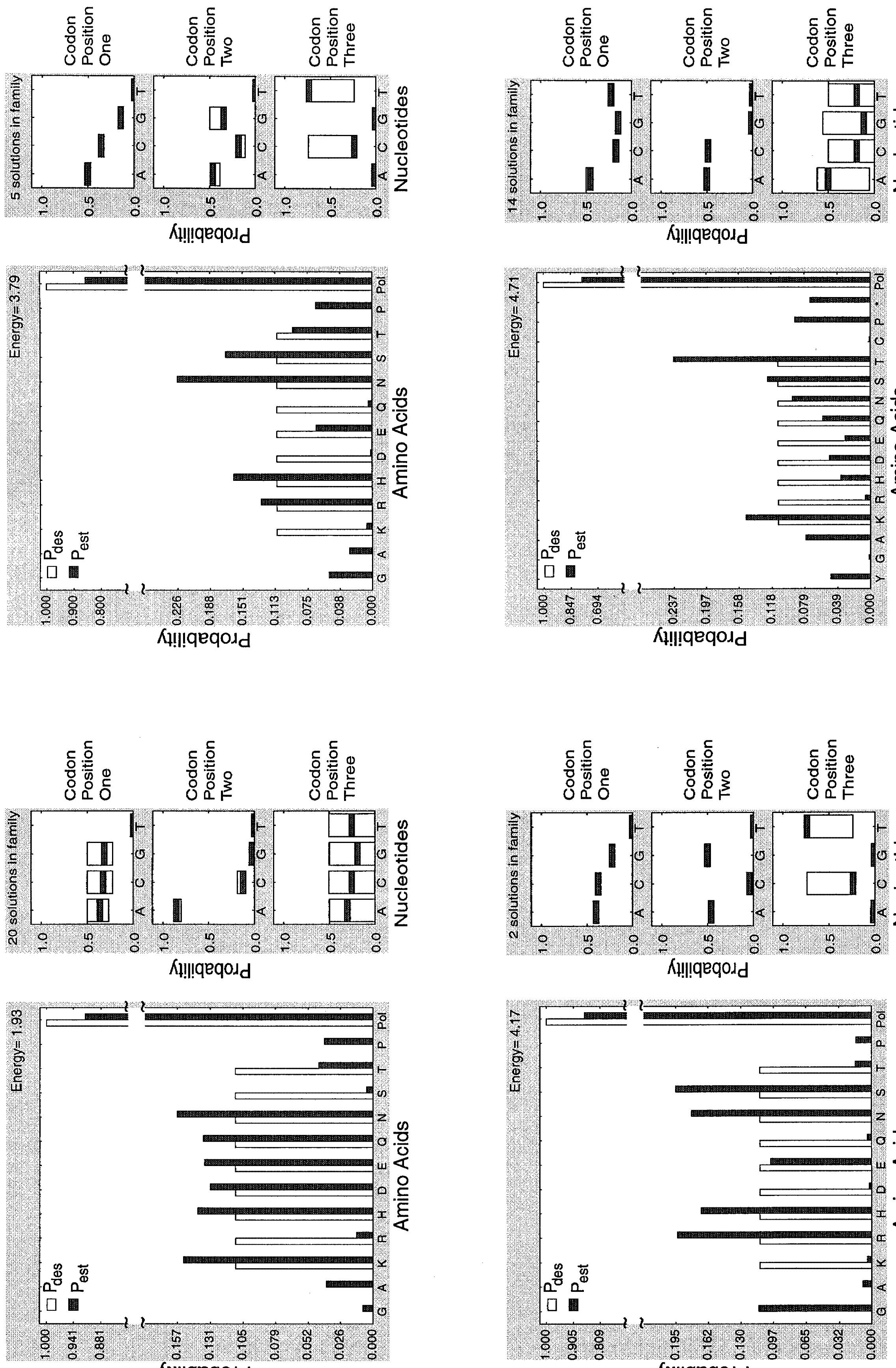

Fig. 4. Families of solutions for polar amino acids, where the probabilities of the rare codons AGG and AGA for arginine were upper bounded by 0.00182 and 0.00287, respectively, corresponding to the frequency of occurrence in *E. coli*.

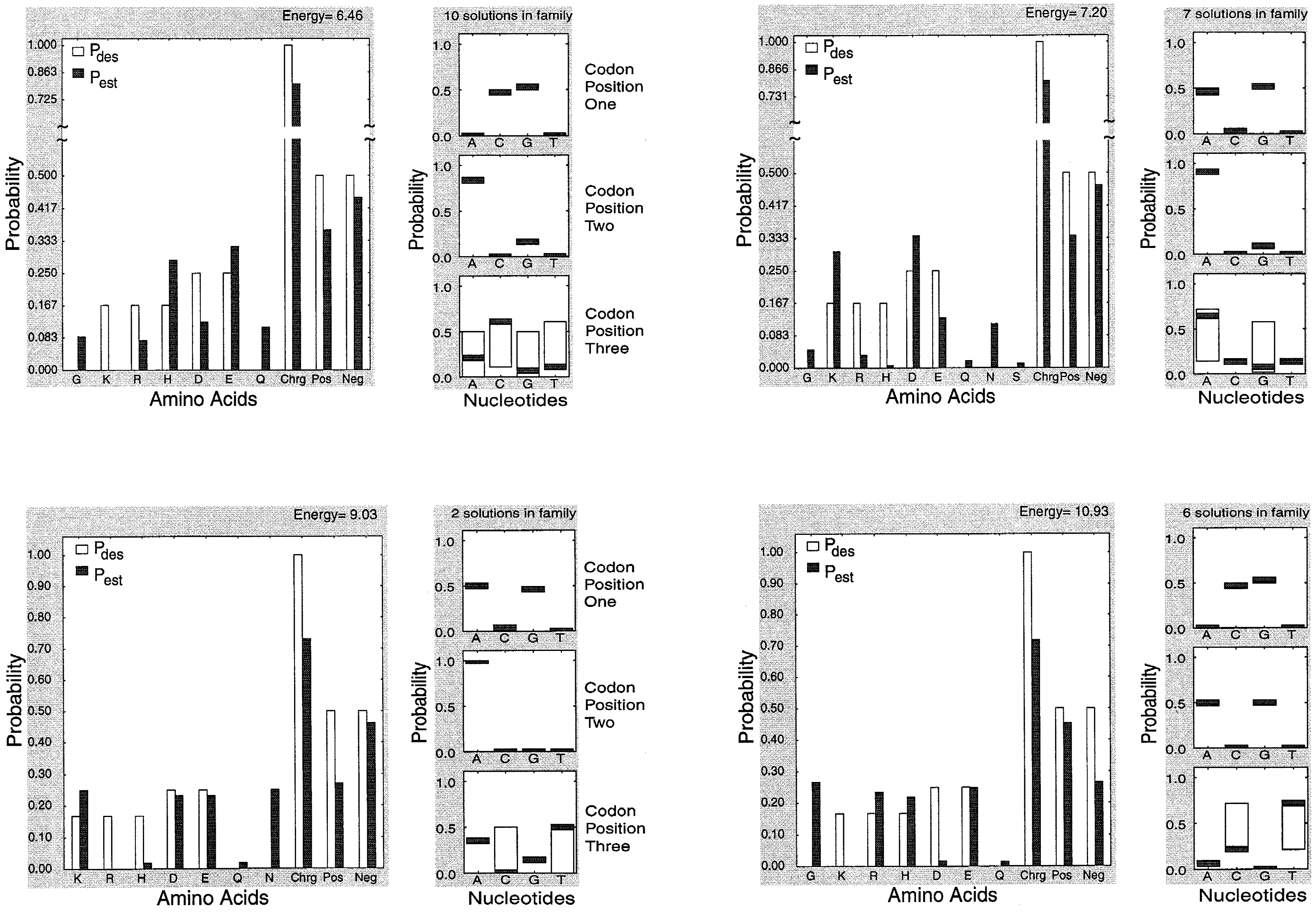

Fig. 5. Families of solutions for charged amino acids, optimized to distribute 1/2 of the probability equally among the positively charged residues, and 1/2 of the probability equally among the negatively charged residues.

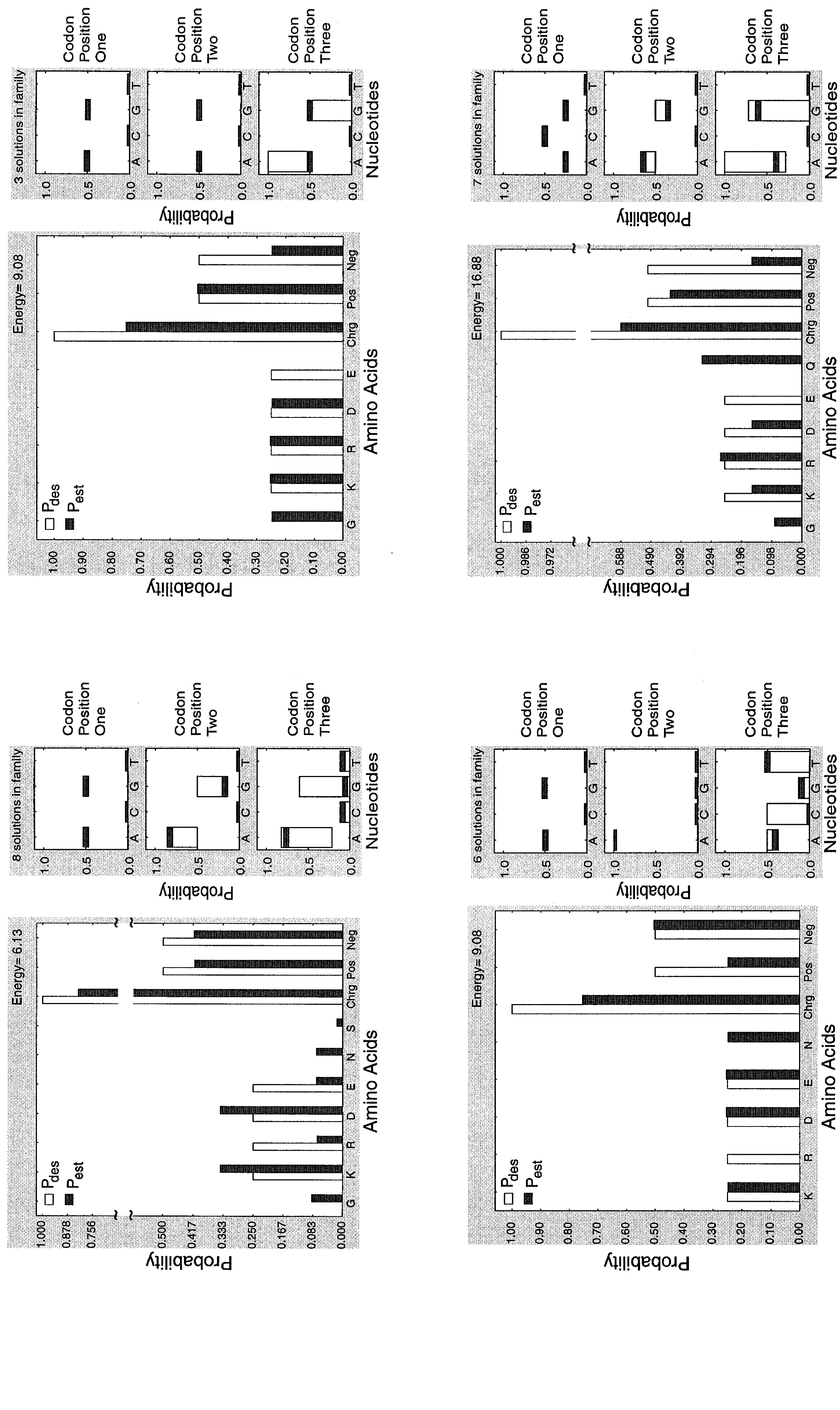

Fig. 6. Families of solutions for charged amino acids, where histidine is not considered charged (since its pK_a is near 7).

Figure 4 demonstrates the effect of placing an upper bound on the probability of the rare codons AGG and AGA for arginine when generating solutions for polar amino acids. The ability to constrain the probability of generating a particular codon could prove useful, as rare codons can often result in decreased levels of protein expression (Kane, 1995). For the arginine example, the two codons were upper bounded by the fraction at which they occur in *Escherichia coli* (0.00182 for AGG and 0.00287 for AGA) (Dalphin et al., 1998). As in Figure 3, the top four families of solutions are shown. Restricting the use of the two rare codons restricts the types of solutions that can be generated: one can notice that two of the examples in Figure 4 have very little probability of generating an arginine due to the avoidance of the rare codons, while all of the examples in Figure 3 have a significant probability of generating arginine.

Figure 5 provides an example of the top four solution types generated when there are competing factors in the desired probability distribution (charged amino acids, for this figure) that cannot by simultaneously optimized. The presence of pairs of amino acids (such as Lys and His) that have only two codons with opposite values for the last base (A and G for Lys, C and T for His) creates a situation where an exact solution is not possible and tradeoffs are required. Generating both His and Lys with nonzero probabilities ensures the noncharged pair Gln and Asn (corresponding to changing the last base in the His and Lys codons) will also be generated with nonzero probability, contrary to the desired probability distribution.

Discussion: For many classes of amino acids (such as tiny, aliphatic, small, and polar), the solutions generated by the program do an excellent job of matching the goal probabilities. Other amino acid classes (such as large, charged, and hydrophobic) demonstrate that, equally often, there are competing factors that make it impossible to simultaneously match the goal probabilities for all of the amino acids. In the examples, amino acids with only two codons proved more likely to lead to problems than other amino acids. The "optimal" solution and tradeoffs depend on the purpose the program user intends for the generated codons.

Hence, in presenting the results from our sample runs we have attempted to provide a diverse selection of potential solutions for each class, which does not lend itself to an exact solution. The base probabilities given in the figures for generating these distributions are of potential use in designing mutational studies and combinatorial libraries. These distributions were chosen for their general applicability. For specific mutational studies where other distributions are required, the program allows for many adjustments of the parameters to best match the user's experimental needs.

Finally, on a practical experimental note, the user needs to decide whether to use a pre-mixed combination of nucleotides, or to have the nucleotide mixture generated by an automated oligonucleotide synthesizer. This decision will be dictated by the specifics of the application. Pre-mixing nucleotides can be expected to give more accurate distributions, although pre-mixing will be less convenient, unless a particular mixture is to be used repeatedly in a given application.

Acknowledgments: Thanks to Masaru Ueno and Yoshihisa Hagihara for their suggestions of how to make the Combinatorial Codons program more intuitive to use. This research was supported by an NIH grant (PO1 HL41484) from the Program of Excellence in Molecular Biology.

References

Dalphin ME, Brown CM, Stockwell PA, Tate WP. 1998. The translational signal database, TransTerm, is now a relational database. *Nuc Acids Res 26*:335–337.

Hu JC, Newell NE, Tidor B, Sauer RT. 1993. Probing the roles of residues at the e and g positions of the GCN4 leucine zipper by combinatorial mutagenesis. *Protein Sci 2*:1072–1084.

Kane JF. 1995. Effects of rare codon clusters on high-level expression of heterologous proteins in *Escherichia coli*. *Curr Opin Biotechnol 6*:494–500.

LaBean TH, Kauffman SA. 1993. Design of synthetic gene libraries encoding random sequence proteins with desired ensemble characteristics. *Protein Sci 2*:1249–1254.

Lawrence CT, Zhou JL, Tits AL. 1997. User's guide for CFSQP version 2.5: A C code for solving (large scale) constrained nonlinear (minimax) optimization problems, generating iterates satisfying all inequality constraints. Technical Report TR-94-16r1. College Park, Maryland: Institute for Systems Research, University of Maryland.

Pu WT, Struhl K. 1993. Dimerization of leucine zippers analyzed by random selection. *Nucleic Acids Research 21*:4348–4355.

Reidhaar-Olson JF, Sauer RT. 1988. Combinatorial cassette mutagenesis as a probe of the informational content of protein sequences. *Science 241*:53–57.

Richards FM. 1974. The interpretation of protein structures: Total volume, group volume distributions and packing density. *J Mol Biol 82*:1–14.

Rose GD, Geselowitz AR, Lesser GJ, Lee RH, Zehfus MH. 1985. Hydrophobicity of amino acid residues in globular proteins. *Science 229*:834–838.

Taylor MW. 1986. Identification of protein sequence homology by consensus template alignment. *J Mol Biol 188*:233–258.

West MW, Hecht MH. 1995. Binary patterning of polar and nonpolar amino acids in the sequences and structures of native proteins. *Protein Sci 4*:2032–2039.