Constrained Codon Optimization by Dynamic Programming

Tuan D. Pham[1,2], James O'Connell[1], and Denis I. Crane[3,4]
[1]School of Computing and Information Technology
[2] Institute for Integrated and Intelligent Systems
[3] School of Biomolecular and Biomedical Science
[4] Eskitis Institute for Cell and Molecular Therapies
Griffith University, Nathan Campus, QLD 4111, Australia
E-mail: t.pham@griffith.edu.au

ABSTRACT

The concept of codon optimization is one that has recently been introduced to overcome the potential functional problems of DNA introduced into foreign hosts. We present a dynamic-programming approach for optimizing the codon selection for a given sequence of amino acids. The proposed approach is very efficient in that its mathematical analysis is easily tractable, its computer implementation is much simpler, and its overall complexity requires much less computational effort in comparison with one of the most recently developed methods for codon optimization.

1. INTRODUCTION

The expression of proteins from DNA introduced into a heterologous host organism is a common approach used for the generation of large quantities of protein for biological research. Bacteria, yeast and baculovirus expression systems are examples of those commonly used to generate proteins of interest from introduced expression vectors or DNA introduced by genome integration.

A potential problem with such approaches is codon usage in different organisms, such that particular codons in the host gene sequence may not be matched with efficient translation of these codons in the organism used for heterologous expression. Optimal codon usage therefore refers to the need to modify the DNA sequence in such a way that codons are selected to maximize the usage in the expression organism while maintaining the host gene's encoded protein sequence [1, 2, 3, 4, 5, 6].

Another important application where DNA sequence changes are necessary is in the use of DNA vaccines [7]. The effectiveness of such vaccines not only relies on the protein expressed but also on the ability of the DNA to elicit an effective immune response. In this respect, it has been demonstrated that certain short DNA motifs may be immuno-stimulatory or immuno-inhibitory and that the modification of vector DNA sequence to optimize motif frequency can enhance immune response and consequently the effectiveness of the vaccine [8, 9].

A pattern matching algorithm for codon optimization and CpG motif engineering in DNA expression vectors has recently been proposed by Satya *et al.* [10]. In this system, a DNA sequence for a given amino acid sequence is synthesized then graph theory is applied to maximize desirable motifs and minimize undesirable motifs. The same authors also surveyed previous related investigations on codon optimization carried out by McInerney [11], Ponger [12], and Lin *et al.* [13].

Given a reasonable number of stages and states, exhaustive enumeration of all solutions along the structured directed graph are impossible. In this paper, we therefore applied the dynamic programming approach [14] for optimizing codon selection and motif frequency.

2. METHOD AND ALGORITHM

Let $(a_1, a_2, \ldots, a_N)$ be a given sequence of amino acids, where N is the size of the sequence. Each a_n, $n = 1, 2, \ldots N$, can represent a valid codon $x_n(k_n)$, $k_n = 1, 2, \ldots, K_n$, where K_n is the number of codons for the amino acid a_n. Two sets of motifs (short DNA sequences whose lengths are between 3 and 8 nucleotides) are also given and designated as desirable and undesirable: the sequence can be modified to maximize and minimize, respectively, the frequency of these two motifs.

The challenge we have set is to find the best sequence of codons, encoding the original host protein sequence, that is optimized for both codon usage in a particular heterologous expression organism and frequency of desirable and undesirable motifs. We can see that this is an optimization problem subjected to the constraints of desirable and undesirable motifs. To obtain an effective solution to this problem, we will cast this problem into the context of dynamic programming [14], in which N is the number of stages (amino acids) and k_n is a state (codon) at stage n. In other words, moving from one stage to the next stage will be equivalent to moving from one amino acid to the next amino acid in the sequence, whereas selecting an optimal state will be equivalent to selecting an optimal codon for the corresponding amino acid.

2.1. Formulation of Gain Matrix

Let $M = 64$ be the number of codons. Also let $\mathbf{T}$, $\mathbf{D}$, and $\mathbf{U}$ be $M \times M$ matrices of which elements represent the number of transitions from one codon to another or to itself based on the codon usage table for a species (http://www.kazusa.or.jp/codon/), the set of desirable motifs, and the set of undesirable motifs, respectively.

The elements $t_{ij} \in \mathbf{T}$ are calculated as

$$t_{ij} = t_i \times t_j \tag{1}$$

where t_i and t_j are the frequencies per thousand of codons i and j given by the codon usage table. Such definition of the matrix $\mathbf{T}$ will have a useful meaning for computing the best path in the context of dynamic programming that will be presented in the following subsections.

The desirable, $\mathbf{D}$, and undesirable, $\mathbf{U}$, matrices are constructed from the lists of desirable and undesirable motifs respectively. Let $\{\mathbf{d}_v, v = 1, 2, \ldots, V\}$ be the set of desirable motifs, and $\mathbf{d}_{vq}, q = 1, 2, \ldots, Q \geq 3$, be a single desirable motif. Similarly, let $\{\mathbf{u}_l, l = 1, 2, \ldots, L\}$ be the set of undesirable motifs, and $\mathbf{u}_{lg}, g = 1, 2, \ldots, G \geq 3$, be a single undesirable motif. We first initialize all elements of $\mathbf{D}$, and $\mathbf{U}$ to be zero. Then the elements of $\mathbf{D}$, and $\mathbf{U}$ will be updated by assigning a weighting value for every occurence of transition from one element to another in each respective motif.

For the desirable-motif transition matrix, we define

$$D_{ij} = \sum_{\sigma \in a_i, a_j} w_\sigma \tag{2}$$

where w_σ is the weighting for a nucleotide σ in an amino acid a_n, and defined as

$$w_\sigma = \frac{1}{|\mathbf{d}_{vq}|}, \tag{3}$$

in which $|\mathbf{d}_{vq}|$ stands for the length of $\mathbf{d}_{vq}$.

Similar procedure is carried out for the construction of the undesirable motif transition matrix. Our next task is to integrate $\mathbf{D}$ and $\mathbf{U}$ into a single matrix that can take into account the conflicting pieces of evidence between the desirable and undesirable condon transitions. This can be done by subtracting the desirable from the undesirable information to obtain a resultant transition matrix, denoted as $\mathbf{R}$, which is defined as

$$\mathbf{R} = \hat{\mathbf{D}} - \hat{\mathbf{U}} \tag{4}$$

where $\hat{\mathbf{D}}$, and $\hat{\mathbf{U}}$ are the normalized versions of $\mathbf{D}$, and $\mathbf{U}$ respectively.

As motifs are short DNA sequences whose lengths generally do not exceed 8 nucleotides, whereas the transition matrix $\mathbf{T}$ is based on the frequencies per thousand triplets, we therefore need to scale down $\mathbf{T}$ in proportion to the number of counts in $\mathbf{R}$. The scaled version of $\mathbf{T}$ is denoted as $\hat{\mathbf{T}}$ and defined by

$$\hat{\mathbf{T}} = \frac{\delta}{1000} \times \mathbf{T} \tag{5}$$

where δ is calculated as the sum of the absolute values of all elements of $\mathbf{R}$ and expressed as follows:

$$\delta = \sum_{i}^{M} \sum_{j}^{M} |r_{ij}|, \; r_{ij} \in \mathbf{R} \tag{6}$$

We are now ready to calculate the gain matrix that takes into account a balanced trade-off of the codon transitions coming from the codon bias information, desirable motifs, and undesirable motifs. This gain matrix is determined as

$$\mathbf{C} = \hat{\mathbf{T}} + \alpha \, \mathbf{R} \tag{7}$$

where α is a positive value that represents a bias for $\mathbf{R}$, and $\mathbf{C}$ is called the gain matrix and will be utilized in the formulation of the dynamic programming.

2.2. Backward Recursive Procedure

Based on the previously defined gain matrix $\mathbf{C} = [c_{ij}]$ (when the subscripts are long as they will be used in the descriptions of the backward and forward algorithms we then express c_{ij} as $c[i, j]$), we denote $f_n(s)$ to be the gain of the path of the amino acids for stages n through N given that the current amino acid is one of the codons in state s. To solve the problem of constrained condon optimization, we need to find the best path for stages 1 through N of which the total gain is maximum. If we let $f_n[s_{n-1}(k_{n-1}), x_n(k_n)]$ be the gain of the best path for stages $n, n+1, \ldots, N$ given that the current amino acid $s_{n-1}(k_{n-1})$, $k_{n-1} = 1, 2, \ldots, K_{n-1}$, where K_{n-1} is the number of states (codons) in state $n-1$, is in state $n-1$, and given that $s_{n-1}(k_{n-1})$ goes to another amino acid $x_n(k_n)$ on stage n. If the gain of the best path from state $x_n(k_n)$ to the final destination on stages $n+1$ through N has already been calculated, then the values $f_n[s_{n-1}(k_{n-1}), x_n(k_n)]$ can be recursively calculated as the gain of going from $s_{n-1}(k_{n-1})$ to $x_n(k_n)$ plus the gain of the best path for stages $n+1$ through N given that the amino acid is in state $x_n(k_n)$.

The above procedure involves a backward computation from the first stage to the last stage of the network. Thus, in order to formulate a general dynamic programming algorithm using the backward recursive procedure for this optimization problem, we present this algorithm in the following four basic steps: initialization, recursion, termination, and back-tracking.

Initialization:

$$f_{N+1}[x_N(k_N)] = 0, k_N = 1, 2, \ldots, N. \tag{8}$$

Recursion:

$$f_n[s_{n-1}(k_{n-1}), x_n(k_n)] = c[s_{n-1}(k_{n-1}), x_n(k_n)] + f_{n+1}[x_n(k_n)], \tag{9}$$

$$\psi[s_{n-1}(k_{n-1})] = \arg\max_{k_n} f_n[s_{n-1}(k_{n-1}), x_n(k_n)], \tag{10}$$

$$\text{where } n = N, N-1, \ldots, 2; \quad k_{n-1} = 1, 2, \ldots, K_{n-1}; \quad k_n = 1, 2, \ldots, K_n.$$

Termination:

$$x_1^*(k_1) = \arg\max_{k_1} f_2[x_1(k_1)], \quad k_1 = 1, 2, \ldots, K_1. \tag{11}$$

Back-tracking:

$$x_n^* = \psi[x_{n-1}^*(k_{n-1})], \quad n = 2, 3, \ldots, N. \tag{12}$$

2.3. Complexity of Computation

The main complexity involves in the calculations of the proposed backward procedure is the computation of the gain function at each stage. Let K_{max} be the maximum number of states (maximum number of codons for an amino acid) and assume that every stage has the same K_{max}. So there are K_{max}^2 additions at each stage, and the overall computation requires on the order of $K_{max}^2\ N$ calculations, where N is the number of stages or the total length of the amino acid sequence. In other words, the complexity of the proposed dynamic-programming approach is less than $K_{max}^2\ N$, whereas the complexity of the multiple-pattern matching method proposed by Satya *et al.* [10] requires on the order of N^2. Our proposed method therefore would make a significant reduction on the computational effort with respect to the length of the amino acid sequence.

3. EXPERIMENTS

For our purposes we chose three randomly generated polypeptides consisting of 100, 200 and 300 amino acids. In each case, we tested the developed algorithm for optimization of codon usage (codon fitness) and the frequency of both desirable and undesirable motifs. All datasets are given as follows, where "-" indicates a stop codon.

Sequence 1 (S1) (100 amino acids):
gcvcrdknpmfqfriikdvawpcpsstmlkpcmfggtkmwipesis
hsddpcwlegdaftrqilyskyemmlimtfiahtqvclkhwciych
ddccmsk-

Sequence 2 (S2) (200 amino acids):
chhiyrsppsdnvctcyhashrctrvvsqwttpcsrnrflkssfpf
pgyslwicwqpvahydnlrvelwasshflqwklcnkeewdyqysah
mhqhyiatyhirkcymddvrytsswdfvrkhldlmmfwmlsrpccp
vkrcamvsserfscflcnewhltlrcslwacrpwhnkkivtwtvsk
kfaitgghlfyhsvg-

Sequence 3 (S3) (300 amino acids):
yfplvslatkpswrekwwkvcitadfepgearkwlktmfqtaqit
mrkheeqhmeqivvnlvtptqhghmrhtnirmkqpsymveyggvr
davegqglcynmhkfnsmpfisqhscpckhvasyysklfphyrk
rgkwqftpdfppinlaahaiqcappaaenciprqclkieqqrlnd
lrvggvftwffacpeteeykhhiindalvwgevfpyqvadtkvrq
heeekvltlllkwagaqqynkepriaksswtiprewnpfmwhqip
qikqtiknmrmslerytrldqidntqyy-

Desirable motifs (23 sequences):
(ctcgac), (ggcgtt), (agcgtt), (gacgtc), (gacgtt), (atcgac), (gtcgtt), (tcgaa), (tcgac), (tcgag), (tcgat), (tcgca), (tcgcc), (tcgcg), (tcgct), (tcgga), (tcggc), (tcggg), (tcggt), (tcgta), (tcgtc), (tcgtg), (tcgtt).

Undesirable motifs (5 sequences):
(ccga), (gcga), (acga), (ccgta), (ccgcg).

The fitness of a codon is the frequency of occurrence of that codon divided by the highest frequency of occurrence for a codon for the same amino acid. The total fitness of a codon sequence, F_T, is defined as the average fitness of all the codons, that is

$$F_T = \frac{1}{N} \sum_{i=1}^{N} F_i \tag{13}$$

where F_i is the fitness of the ith codon, and is defined as the ratio of the frequency of the ith codon, ω_i, of the corresponding amino acid to that of the best codon, ω_b, of the same amino acid [10]

$$F_i = \frac{\omega_i}{\omega_b} \tag{14}$$

in which ω_b is the maximum frequency assigned to a codon in the set of codons of the amino acid.

Table 1 shows the results of the fitness values, desirable motifs, and undesirable motifs in three cases when the optimization is subjected to maximizing desirable motifs, minimizing undesirable motifs, and both maximizing and minimizing desirable and undesirable motifs simultaneously. We also studied the sensitivity of the bias parameter α defined in (7). A convergence for the total fitness is when $\alpha = 10$, for desirable motifs when $\alpha = 4$, and undesirable motifs when $\alpha = 1$.

Overall, the results obtained confirmed that the developed algorithms were effective in optimizing both codon fitnesss and frequency of selected motifs for the different length amino acid sequences. However the analyses did show that with the shortest length sequence (100 amino acids) there was a limit to the effectiveness of these manipulations, suggesting, as would be expected, that there is a lower limit to the number of codon choices in any sequence. Also as expected, the analyses indicated that increasing α led

Table 1. Results on constrained codon optimization

$\alpha = 0.5$									
	Maximize DMs			Minimize UMs			Maximize DMs and minimize UMs		
	S1	S2	S3	S1	S2	S3	S1	S2	S3
Total fitness	0.9140	0.8817	0.9532	0.9836	0.9930	0.9897	0.9144	0.8925	0.9529
Desirable motifs (DMs)	12	38	30	2	4	7	12	35	27
Undesirable motifs (UMs)	8	4	12	0	0	0	1	0	2
$\alpha = 1$									
	Maximize DMs			Minimize UMs			Maximize DMs and minimize UMs		
	S1	S2	S3	S1	S2	S3	S1	S2	S3
Total fitness	0.8989	0.8421	0.9244	0.9836	0.9930	0.9897	0.9022	0.8788	0.9402
Desirable motifs (DMs)	15	45	42	2	4	7	14	38	33
Undesirable motifs (UMs)	8	5	14	0	0	0	0	0	0
$\alpha = 2$									
	Maximize DMs			Minimize UMs			Maximize DMs and minimize UMs		
	S1	S2	S3	S1	S2	S3	S1	S2	S3
Total fitness	0.8872	0.8061	0.8738	0.9836	0.9930	0.9897	0.8962	0.8480	0.9078
Desirable motifs (DMs)	17	46	48	2	4	7	14	45	45
Undesirable motifs (UMs)	8	6	16	0	0	0	0	0	0

to a decrease in codon fitness and an increase in the number of desirable motifs that could be included. The results also indicated that an inadvertent consequence of maximising the number of desirable motifs was an increase in that the number of undesirable motifs, whereas the minimisation of undesirable motifs led to an increase in codon fitness and loss of desirable motifs.

In general, maximising desirable motifs while minimising undesirable motifs produced the best compromise for the fitness value, desirable motif expression, and undesirable motif suppression. Furthermore, the results also clearly demonstrated that the choice of algorithm will be determined by the outcome required of the analysis - whether this be codon fitness, maximised content of desirable motifs, minimised content of undesirable motifs, or a combination of these factors. Direct comparison with the codon optimization method developed by Satya *et al.* [10] was difficult because the test data used by the latter were not made available. However, our method is much easier for computer implementation, and its computation is more efficient as its complexity is approximately on the linear order of the sequence length (N), whereas the latter is on the order of N^2.

4. CONCLUSIONS

In the foregoing sections we have discussed and presented a dynamic-programming approach known as the backward and forward recursive procedures for codon optimization subjected to the constraints of the codon bias, and immuno-modulatory motif control. To proceed with the dynamic programming procedures, we have also formulated the so-called gain matrix that makes a compatible fusion of codon usage information, desirable motifs, and undesirable motifs and is used as a scoring criteria for choosing the best path in the network. The gain function, which is inversely equivalent to the cost function of many dynamic programming methods, is essential for selecting the best path of codons and can be further studied by modifying the gain matrix without changing the formulation of the backward or the forward procedure. The proposed algorithms are simple for computer implementation as well as effective for real-time computation.

References

[1] C.H. Kim, Y. Oh, and T.H. Lee, Codon optimization for high level expression of human erythropoietin (EPO) in mammalian cells, *Gene*, 199 (1997) 293-301.

[2] L. Wu L, M.A. Barry, Fusion protein vectors to increase protein production and evaluate the immunogenicity of genetic vaccines, *Mol. Ther.* 2 (2000), pp. 288-297.

[3] M.L. Valencik, and J.A. McDonald, Codon optimization markedly improves doxycycline regulated gene expression in the mouse heart, *Transgenic Res.*, 10 (2001), pp. 269-275.

[4] M. Fuller, and D.S.A. Anson , Helper plasmids for production of HIV-1-derived vectors, *Hum. Gene. Ther.*, 12 (2001), pp. 2081-2093.

[5] A. Cid-Arregui, V. Juarez, and H. zur Hausen, A synthetic E7 gene of human papillomavirus type 16 that yields enhanced expression of the protein in mammalian cells and is useful for DNA immunization studies, *J. Virol*, 77 (2003), pp. 4928-4937.

[6] E.M. Slimko, and H.A. Lester, Codon optimization of Caenorhabditis elegans GluCl ion channel genes for mammalian cells dramatically improves expression levels, *J. Neirosci. Meth.*, 124 (2003), pp. 75-81.

[7] S. Gurunathan, D.M. Klinman, and R.A. Seder, DNA vaccines: immunology, application, and optimization, *Annu. Rev. Immunol.*, 18 (2000) 927-974.

[8] A.M. Krieg, CpG motifs in bacterial DNA and their immune effects, *Annu. Rev. Immunol.*, 20 (2002) 709-760.

[9] A.M. Krieg , A.K. Yi, J. Schoor, and H.L. Davis, The role of CpG dinucleotides in DNA vaccines, *Trends Microbiol.*, 6 (1998) 23-27.

[10] R.V. Satya, A. Mukherjee, and U. Ranga, A pattern matching algorithm for codon optimization and CpG motif-engineering in DNA expression vectors, *Proc. Computational Systems Bioinformatics (CSB'03)*, Stanford, USA, 2003, pp. 294-305.

[11] J.O. McInerney, GCUA: General codon usage analysis, *Bioinformatics*, 1998 (14) 372-373.

[12] L. Ponger, and D. Mouchiroud, CpGProD: Identifying CpG islands associated transcription start sites in large genomic mammalian sequences, *Bioinformatics*, 2002 (18) 631-633.

[13] Y.L. Lin, X. Huang, T. Jiang, and K.M. Chao, MAVG: Locating non-overlapping maximum average segments in a given sequence, *Bioinformatics*, 19 (2003), 151-152.

[14] R. Bellman, and S. Dreyfus. *Applied Dynamic Programming*. Princeton University Press, Princeton, 1962.