Efficient codon optimization with motif engineering

Anne Condon and Chris Thachuk

Department of Computer Science, University of British Columbia,
Vancouver, BC, Canada V6T 1Z4
{condon,cthachuk}@cs.ubc.ca

Abstract. It is now common to add protein coding genes into cloning vectors for expression within non-native host organisms. Codon optimization supports translational efficiency of the desired protein product, by exchanging codons which are rarely found in the host organism with more frequently observed codons. Motif engineering, such as removal of restriction enzyme recognition sites or addition of immuno-stimulatory elements, is also often necessary. We present an algorithm for optimizing codon bias of a gene with respect to a well motivated measure of bias, while simultaneously performing motif engineering. The measure is the previously studied codon adaptation index, which favors the use, in the gene to be optimized, of the most abundant codons found in the host genome. We demonstrate the efficiency and effectiveness of our algorithm on the GENCODE dataset and provide a guarantee that the solution found is always optimal.

1 Introduction

Gene synthesis is now an economical and technically viable option for the construction of non-natural genes. Synthetic genes can be novel or derivatives of those found in nature. In either case, the expression levels of these genes, when inserted into the genome of a host organism, depend on many factors. One important factor is the bias of codon usage, relative to the host organism [16, 7, 10]. Note that for each amino acid in a protein, there may be many (up to six) valid codons, as given by the genetic code. Loosely speaking, the codon bias of a gene for the protein measures how well – or poorly – codons used in the gene match codon usage in the genome of a host organism (we describe specific measures later in this paper). Several studies have indicated that [10, 3, 11] codon optimization is necessary to ensure designed genes are maximally expressed within the host.

In addition to optimizing the codon bias of a gene relative to the genome of a host, it is often desirable to add or remove certain motifs via silent mutation, whereby DNA sequence is altered without changing the expressed amino acid sequence. Removal or addition of motifs can be treated as optimization criteria to be minimized or maximized. For example, with immuno-regulatory CpG

motifs in mammalian expression vectors [13] it is desirable to minimize immuno-inhibitory elements and maximize immuno-stimulatory motifs. In the remainder of this work, we will refer to inclusion or exclusion of motifs, via silent mutation, as motif engineering.

A number of published software tools are capable of codon optimization, including DNA Works [8], Codon Optimizer [2], GeMS [9], Gene Designer [17], JCat [4], OPTIMIZER [12], the Synthetic Gene Designer [18], UpGene [3] and a method by Satya *et. al* [13]. Some of these methods also consider the other problem considered here, motif engineering. Of these, only the method of Satya *et. al* provides a mathematical guarantee of finding an optimal solution when one exists. However, their method – based on the graph theoretic approach of finding a critical path – runs in $O(n^2)$ time and space, where n is the length of the DNA sequence being optimized. In this work, we propose the first linear time and space codon optimization algorithm, which is guaranteed to find an optimal solution that also satisfies motif engineering constraints. We have focused our attention on optimizing codon usage according to the Codon Adaptation Index (CAI). The index, originally proposed by Sharp and Li [14], is based on the premise of each amino acid having a 'best' codon for a particular organism. This perspective evolved from the observation that protein expression is higher in genes using codons of high fitness and lower in genes using rare codons [6]. It is believed that this is due to the relative availability of tRNAs within a cell.

We also provide an experimental study of the performance of our algorithm on a biological data set comprising 3,157 coding sequence regions of the GENECODE subset of the ENCODE dataset [15].

The remainder of this paper is structured as follows. In the Preliminaries section, we formally define the problem of codon optimization. We detail the general objectives of the problem, and formalize the goals of motif engineering. We then present our algorithm, providing a proof of correctness and time and space analysis. In the Empirical Results section, we describe the performance of our algorithm, both in terms of run-time efficiency and also in terms of the quality of optimization achieved. Finally, we conclude with a summary of our major findings and directions for future work.

2 Preliminaries

A DNA strand is a string over the alphabet of DNA. A *codon* is a triple over the DNA alphabet and therefore there are at most $4^3 = 64$ distinct codons. An *amino acid sequence* is a string over the alphabet of amino acids, $\Sigma_{AA} = \{Ala, Arg, Asn, \ldots, Tyr, Val, stop\}$, with each symbol representing an amino acid and the special symbol '*stop*' denoting a string terminal. We assume there is a predetermined ordering of amino acids, for example, lexicographic.

Therefore, we can represent an amino acid sequence A as a sequence of integers, with $A = \alpha_1, \alpha_2, \ldots, \alpha_{|A|}$, where $1 \leq \alpha_k \leq 21$, for $1 \leq k \leq |A|$. We denote the i^{th} amino acid by $\lambda(i)$. The *genetic code* is a mapping between amino acids and codons. However, as there are 64 possible codons and only 20 amino

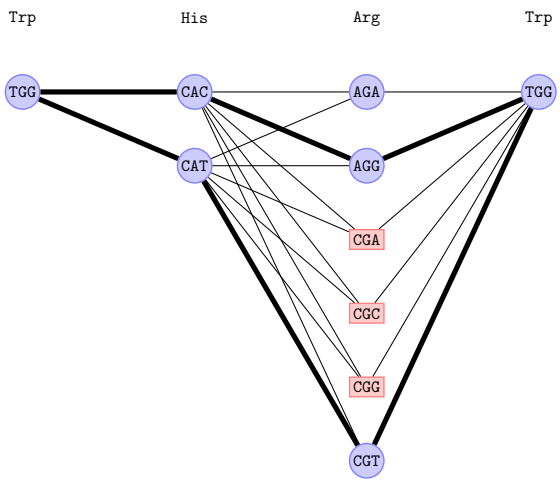

Fig. 1. Shown above is an instance consisting of four amino acids, a forbidden set $\mathcal{F} = \{$ `CGC`, `CGA`, `CGG`, `ACC`, `TAG` $\}$, a desired set $\mathcal{D} = \{$ `TTG`, `GGT`, `CCG` $\}$, and therefore $k = 1$. Arginine has six corresponding codons, however, three of them appear in $\mathcal{F}$ and are shown with red boxes. A valid codon assignment for this instance must contain no occurrence of a forbidden motif and one occurrence of a desired motif. There are two valid codon assignments for this problem instance, shown as paths with bold edges. The top (bottom) path denotes an assignment containing the desired motif `GGT` (`TTG`).

acids (plus one stop symbol), the code is degenerate, resulting in a one-to-many mapping from each amino acid to a set of corresponding codons.

We define $|\lambda(i)|$ to be the number of codons corresponding to the i$^{\text{th}}$ amino acid and $\lambda_j(i)$ to be the j^{th} such codon, $1 \leq j \leq |\lambda(i)|$, where again we use lexicographic ordering. Therefore, we can define a *codon design*, with respect to an amino acid sequence, as a sequence of codon indices. Again consider the problem instance in Figure 1. For the Arginine amino acid (`Arg`) which is the second amino acid in lexicographic order, $|\lambda(2)| = 6$ and $\lambda_3(2)$ is the codon `CGA`. The DNA sequence `TGA CAC CGA TGG` can be represented by the codon index sequence $S = 1, 1, 3, 1$.

A *codon's frequency* is the number of times that it appears in nature, divided by the total number of times that all codons corresponding to the same amino acid appear in nature. By "in nature", we mean codon frequencies present in some reference sequence or set of sequences such as a genome or set of genomes. As an example, if for some amino acid index i, $|\lambda(i)| = 2$, and the codon $\lambda_1(i)$ is observed 37 times in nature, while $\lambda_2(i)$ is observed 63 times, we can define the frequency of $\lambda_1(i)$ to be $\frac{37}{37+63} = 0.37$. Let $\rho_j(i)$ denote the frequency of the j^{th} codon of the i$^{\text{th}}$ amino acid, $1 \leq j \leq |\lambda(i)|$. Note that $\sum_{j=1}^{|\lambda(i)|} \rho_j(i) = 1.0$, for any i, assuming $\lambda(i)$ is in the reference set. In the example above, we say that $\lambda_2(i)$

is the *most frequent codon*. Note that it is possible for more than one codon to have this property.

A *codon's fitness* is the number of times that it appears in nature, divided by the number of occurrences of the corresponding most frequent codon (originally referred to as the relative adaptiveness of a codon [14]). Let $\tau_j(i)$ denote the fitness value of the j^{th} codon of the i^{th} amino acid. Returning to our previous example, if the i^{th} amino acid has two codons with frequencies $\rho_1(i) = 0.37$ and $\rho_2(i) = 0.63$, then their fitness values, denoted by $\tau_1(i)$ and $\tau_2(i)$ respectively, are $0.37/0.63 \approx 0.59$ and $0.63/0.63 = 1.0$. Note that a most frequent codon will always have a fitness value of 1.0.

Motif engineering We focus our attention on designing codon sequences which minimize occurrences of forbidden motifs from a predetermined set, $\mathcal{F}$, while maximizing occurrences of desired motifs from a predetermined set, $\mathcal{D}$. A codon design – a sequence of codon assignments – is said to be *valid* with respect to an amino acid sequence it codes for if it satisfies the following constraints, in order: the DNA sequence corresponding to the codon design (1) contains the minimum possible number of forbidden motifs, and (2) contains the maximum possible number of desired motifs, given that (1) is satisfied. It is important to recognize that a valid design does not necessarily guarantee that the number of occurrences of desired motifs is the maximum number possible, of all possible codon designs. Again, consider the problem instance of Figure 1. Two codon designs result in a minimum number of forbidden motif occurrences (none), shown with paths having bold edges. Both of these paths also contain one occurrence of a desired motif. The top (bottom) path denotes an assignment containing the desired motif `GGT` (`TTG`). Therefore, a valid codon design for this instance, by our previous definition, contains no forbidden motifs and one desired motif. Notice that the DNA sequence `TGG CAC CGG TGG`, corresponding to a codon design $S = 1, 1, 5, 1$, actually contains more desired motifs (two) than a valid codon design; however, it does contain one forbidden motif and therefore cannot be valid.

We now develop some notation for motif engineering. For a sequence of amino acid indices $A = \alpha_1, \alpha_2, \ldots, \alpha_{|A|}$, a corresponding codon design $S = c_1, c_2, \ldots, c_{|A|}$, a set of forbidden motifs $\mathcal{F}$ and a set of desired motifs $\mathcal{D}$, let $M_{\mathcal{F}}(\lambda_{c_i}(\alpha_i) \ldots \lambda_{c_j}(\alpha_j))$ and $M_{\mathcal{D}}(\lambda_{c_i}(\alpha_i) \ldots \lambda_{c_j}(\alpha_j))$ be the number of occurrences of forbidden motifs and desired motifs, respectively, in the DNA sequence $\lambda_{c_i}(\alpha_i) \ldots \lambda_{c_j}(\alpha_j)$, where $j \geq i$. For convenience in our algorithms, we also introduce $M'_{\mathcal{F}}(\lambda_{c_i}(\alpha_i) \ldots \lambda_{c_j}(\alpha_j))$ and $M'_{\mathcal{D}}(\lambda_{c_i}(\alpha_i) \ldots \lambda_{c_j}(\alpha_j))$ which respectively determine the number of forbidden and desired motifs in $\lambda_{c_i}(\alpha_i) \ldots \lambda_{c_j}(\alpha_j)$ that end within the last codon position (the last 3 bases), here indexed by j. For instance consider the codon design $S = 1, 1, 5, 1$ of the problem instance in Figure 1. $M_{\mathcal{D}}(\texttt{TGGCACCGGTGG}) = 2$ as it contains the motifs `CCG` and `GGT`, however, $M'_{\mathcal{D}}(\texttt{TGGCACCGGTGG}) = 1$ as only the motif `GGT` ends within the last codon.

In practice, forbidden and desired motifs are short and we assume their length is bounded by a constant, g [13].

Observation 1 *If the largest forbidden or desired motif is of length g, then any forbidden or desired motif can span at most $k+1$ consecutive codons, where $k = \lceil g/3 \rceil$.*

Codon optimization The codon adaption index (CAI) is a metric defined in terms of the relative fitness of codons constituting a codon design. For some codon design $S = c_1, c_2, \ldots, c_{|A|}$, which correctly codes for a desired amino acid sequence $A = \alpha_1, \alpha_2, \ldots, \alpha_{|A|}$, the CAI value for S with respect to A, CAI(S, A), can be calculated as in Eqn. (1). Based on this definition, if S consists only of most frequent codons, it would have a CAI value of 1.0. Intuitively, the higher the CAI value, the better.

$$\text{CAI}(S, A) = \left(\prod_{i=1}^{|A|} \tau_{c_i}(\alpha_i) \right)^{\frac{1}{|A|}} \tag{1}$$

With the previously defined definitions, notation, and optimization criteria, we now formally define the problem of codon optimization with motif engineering.

The CAI codon optimization problem with motif engineering
Instance: Amino acid sequence represented by the sequence of indices $A = \alpha_1, \alpha_2, \ldots, \alpha_{|A|}$, a set of forbidden motifs $\mathcal{F}$, and a set of desired motifs $\mathcal{D}$.

Problem: Find a codon design S^*, with $|S^*| = |A|$, corresponding to A such that S^* is *valid*, with respect to $\mathcal{F}$ and $\mathcal{D}$, and $CAI(S^*, A) = \max\{CAI(S, A) | S \in \mathbf{S}(A)\}$, where $\mathbf{S}(A)$ is the set of all valid codon designs corresponding to A. S^* is an *optimal codon design with respect to the CAI measure.*

3 A DP algorithm for CAI optimization

We now propose a linear time and space dynamic programming algorithm guaranteed to maximize the CAI measure, such that the codon design is *valid*. In terms of efficiency, this is a direct improvement in both run-time and space over the current state-of-the-art, previously proposed by Satya *et al.* [13]. Although we have chosen to first ensure forbidden motifs are minimized, then desired motifs maximized and finally the CAI value maximized, it should be clear that the algorithm we present can be adapted to optimize these criteria in any order.

One necessary feature of a codon optimization algorithm is an efficient means to detect if a forbidden motif from $\mathcal{F}$, or a desired motif from $\mathcal{D}$, is present in a potential design. For both algorithms proposed in this work, we utilize an Aho-Corasick search for this purpose. Briefly, the Aho-Corasick algorithm builds a keyword tree (trie) for $\mathcal{F}$ and transforms the structure into an automaton with the addition of failure links. Space and time complexity for building the initial structure is $O(h)$, where h is the sum of the lengths of the motifs in $\mathcal{F}$. Queries

to determine if a sequence b contains any forbidden motif take $O(|b|)$ time [1]. Likewise, a second tree is constructed for the desired motifs in $\mathcal{D}$. For a detailed description of the algorithm and existing applications of its use in computational biology, see Gusfield [5]. We note that Satya *et. al* [13] use the same approach for motif detection in their $\theta(n^2)$ algorithm. We note that any dictionary matching algorithm can be employed for the same task; however, Aho-Corasick automata were chosen due to their simpler implementation.

We first define three quantities that will be important in describing our algorithm. The first quantity, $F^i_{c_{i-k+1},\ldots,c_{i-1},c_i}$, denotes the minimum possible number of forbidden motifs in a DNA sequence which codes for an amino acid sequence $A = \alpha_1, \alpha_2, \ldots, \alpha_i$, given that the last k codons (of i total codons) have indices denoted as $c_{i-k+1}, \ldots, c_{i-1}, c_i$. Similarly, the second quantity, $D^i_{c_{i-k+1},\ldots,c_{i-1},c_i}$, denotes the maximum possible number of desired motifs, among those sequences which contain a minimum number of forbidden motifs. $P^i_{c_{i-k+1},\ldots,c_{i-1},c_i}$ denotes the maximum possible CAI score among all valid sequences.

Our algorithm stores a k-dimensional entry for each position i, $k \leq i \leq |A|$, of the input amino acid sequence, where $k = \lceil g/3 \rceil$ and g is the constant bounding the length of any forbidden or desired motif. The base case occurs when $i = k$ and is computed as follows. Every combination of codons for the first k amino acids is evaluated to determine, independently, the number of forbidden and desired motifs fully contained within the k consecutive codons (Eqn. (2) and Eqn. (3), respectively) and the CAI value (Eqn. (4)).

$$F^k_{c_1,c_2,\ldots,c_{k-1},c_k} = M_{\mathcal{F}}\left(\lambda_{c_1}(\alpha_1)\lambda_{c_2}(\alpha_2)\ldots\lambda_{c_{k-1}}(\alpha_{k-1})\lambda_{c_k}(\alpha_k)\right) \tag{2}$$

$$D^k_{c_1,c_2,\ldots,c_{k-1},c_k} = M_{\mathcal{D}}\left(\lambda_{c_1}(\alpha_1)\lambda_{c_2}(\alpha_2)\ldots\lambda_{c_{k-1}}(\alpha_{k-1})\lambda_{c_k}(\alpha_k)\right) \tag{3}$$

$$P^k_{c_1,c_2,\ldots,c_{k-1},c_k} = \prod_{i=1}^{k}\left(\tau_{c_i}(\alpha_i)\right) \tag{4}$$

The recursive case occurs for $i > k$. By Observation 1, a forbidden motif could span $k+1$ codons. Therefore, it is necessary to evaluate the last $k+1$ codons of a potential design to ensure codons are selected which 1) minimize forbidden motifs, then 2) maximize desired motifs, then 3) maximize the CAI score.

For any arbitrary assignment of the last k codons, we select the codon preceding them, denoted by the index c_{i-k}, such that the sum of forbidden motifs ending at position $i-1$, $F^{i-1}_{c_{i-k},\ldots,c_{i-2},c_{i-1}}$, and the count of new forbidden motifs which end in the new codon c_i, determined by the function $M'_{\mathcal{F}}$, is minimized. The number of forbidden motifs is recorded.

$$F^i_{c_{i-k+1},\ldots,c_{i-1},c_i} = \min_{1\leq c_{i-k}\leq|\lambda(\alpha_{i-k})|} \left\{F^{i-1}_{c_{i-k},\ldots,c_{i-2},c_{i-1}} + M'_{\mathcal{F}}\left(\lambda_{c_{i-k}}(\alpha_{i-k})\ldots\lambda_{c_{i-1}}(\alpha_{i-1})\lambda_{c_i}(\alpha_i)\right)\right\} \tag{5}$$

Similarly, D is calculated in the same manner, after ensuring that the minimal number of forbidden motifs criteria is first satisfied.

$D^i_{c_{i-k+1},\ldots,c_{i-1},c_i} = \max_{1\le c_{i-k}\le|\lambda(\alpha_{i-k})|}$

$$\left\{\begin{array}{ll} -\infty & \text{, if } F^{i-1}_{c_{i-k},\ldots,c_{i-2},c_{i-1}} + \\ & M'_{\mathcal{F}}\left(\lambda_{c_{i-k}}(\alpha_{i-k})\ldots\lambda_{c_{i-1}}(\alpha_{i-1})\lambda_{c_i}(\alpha_i)\right) \\ & \neq F^i_{c_{i-k+1},\ldots,c_{i-1},c_i} \\ D^{i-1}_{c_{i-k},\ldots,c_{i-2},c_{i-1}} & \\ +M'_{\mathcal{D}}\left(\lambda_{c_{i-k}}(\alpha_{i-k})\ldots\lambda_{c_i}(\alpha_i)\right) & \text{, otherwise} \end{array}\right\} \tag{6}$$

Likewise, P is calculated to first ensure forbidden motifs are minimized, followed by desired motifs being maximized. Of these possible codon assignments, the one with the highest CAI value is selected and the score recorded.
$P^i_{c_{i-k+1},\ldots,c_{i-1},c_i} = \max_{1\le c_{i-k}\le|\lambda(\alpha_{i-k})|}$

$$\left\{\begin{array}{ll} -\infty & \text{, if } F^{i-1}_{c_{i-k},\ldots,c_{i-2},c_{i-1}} + \\ & M'_{\mathcal{F}}\left(\lambda_{c_{i-k}}(\alpha_{i-k})\ldots\lambda_{c_i}(\alpha_i)\right) \neq F^i_{c_{i-k+1},\ldots,c_{i-1},c_i} \\ & \vee D^{i-1}_{c_{i-k},\ldots,c_{i-2},c_{i-1}} \\ & +M'_{\mathcal{D}}\left(\lambda_{c_{i-k}}(\alpha_{i-k})\ldots\lambda_{c_i}(\alpha_i)\right) \neq D^i_{c_{i-k+1},\ldots,c_{i-1},c_i} \\ \tau_{c_i}(\alpha_i)\times P^{i-1}_{c_{i-k},\ldots,c_{i-2},c_{i-1}} & \text{, otherwise} \end{array}\right\} \tag{7}$$

Eqn. (10) determines the optimal CAI score up to position i of the input amino acid sequence. Therefore, the optimal CAI value of some input sequence A of length $|A|$ is given by $\widetilde{P^{|A|}_k}$, where

$$\widetilde{F^i_k} = \min_{\substack{1\le c_i\le|\lambda(\alpha_i)|\\ 1\le c_{i-1}\le|\lambda(\alpha_{i-1})|\\ \vdots \\ 1\le c_{i-k+1}\le|\lambda(\alpha_{i-k+1})|}} \left\{F^i_{c_{i-k+1},\ldots,c_{i-1},c_i}\right\} \tag{8}$$

$$\widetilde{D^i_k} = \max_{\substack{1\le c_i\le|\lambda(\alpha_i)|\\ 1\le c_{i-1}\le|\lambda(\alpha_{i-1})|\\ \vdots \\ 1\le c_{i-k+1}\le|\lambda(\alpha_{i-k+1})|}} \left\{\begin{array}{ll} D^i_{c_{i-k+1},\ldots,c_{i-1},c_i} & \text{, if } F^i_{c_{i-k+1},\ldots,c_{i-1},c_i} = \widetilde{F^i_k} \\ -\infty & \text{, otherwise} \end{array}\right\} \tag{9}$$

$$\widetilde{P^i_k} = \max_{\substack{1\le c_i\le|\lambda(\alpha_i)|\\ 1\le c_{i-1}\le|\lambda(\alpha_{i-1})|\\ \vdots \\ 1\le c_{i-k+1}\le|\lambda(\alpha_{i-k+1})|}} \left\{\begin{array}{ll} P^i_{c_{i-k+1},\ldots,c_{i-1},c_i} & \text{, if } F^i_{c_{i-k+1},\ldots,c_{i-1},c_i} = \widetilde{F^i_k} \\ & \wedge D^i_{c_{i-k+1},\ldots,c_{i-1},c_i} = \widetilde{D^i_k} \\ -\infty & \text{, otherwise} \end{array}\right\} \tag{10}$$

The correctness of the algorithm can be shown by induction on the position in the amino acid sequence. Lemma 1 shows that Eqn. (7) gives an optimal score

under the assumption that the previous k codons are fixed. Since Eqn. (10) evaluates all combinations of the previous k codons, Theorem 1 states that an optimal design must be found, if one exists.

Lemma 1. *$P^i_{c_{i-k+1},\ldots,c_{i-1},c_i}$ of Eqn. (7) correctly determines the score of the optimal valid codon design up to the i^{th} codon position, having the codon assignment $c_{i-k+1},\ldots,c_{i-1},c_i$ for the last k codons, given that the maximum length of any motif is $3k$.*

Proof. We will argue by induction. The base case ($i = k$) is trivially *valid* as Eqn. (4) correctly determines the CAI score of the first k codons, by definition.

Assume $P^{i-1}_{c'_{i-k},\ldots,c'_{i-2},c'_{i-1}}$ correctly determines the score of an optimal *valid* codon assignment, up to position $i-1$, having the codon assignment $c'_{i-k},\ldots,c'_{i-2}$, c'_{i-1} for the last k codons. Similarly, assume F^{i-1} and D^{i-1} are also correct for the corresponding codon assignment. When moving one position ahead, from $i-1$ to i, we must consider the case of any new motifs we may introduce. By Observation 1, any new motif which ends within codon c_i could not extend past codon c_{i-k}. There are at most 6 possible codon assignments to position c_{i-k} that can directly precede a specific codon assignment $c_{i-k+1},\ldots,c_{i-1},c_i$ ending at position i as there are at most 6 codons for any amino acid. Therefore, the optimal assignment(s) to c_{i-k} must be a subset of these possibilities. $M'_{\mathcal{F}}\left(\lambda_{c_{i-k}}(\alpha_{i-k})\ldots\lambda_{c_{i-1}}(\alpha_{i-1})\lambda_{c_i}(\alpha_i)\right)$ calculates the number of new forbidden motifs introduced in the codon assignment $c_{i-k},\ldots,c_{i-1},c_i$ which end in codon c_i. By our assumption, $F^{i-1}_{c_{i-k},\ldots,c_{i-2},c_{i-1}}$ correctly determines the minimum number of forbidden motifs having codon assignment $c_{i-k},\ldots,c_{i-2},c_{i-1}$, ending at position $i-1$. Therefore, the sum of these two quantities correctly determines the minimum number of forbidden motifs. As codons $c_{i-k+1},\ldots,c_{i-1},c_i$ are fixed, and Eqn. (5) evaluates every possible assignment to c_{i-k} to determine a minimum, then it must be the case that $F^i_{c_{i-k+1},\ldots,c_{i-1},c_i}$ is the minimum number of forbidden motifs up to position i, having the codon assignment $c_{i-k+1},\ldots,c_{i-1},c_i$ for the last k codons. We argue similarly for $D^i_{c_{i-k+1},\ldots,c_{i-1},c_i}$ in Eqn. (6) with the addition that any assignment of c_{i-k} also be forbidden motif minimal ensured by line 1 of the equation.

Finally, consider $P^i_{c_{i-k+1},\ldots,c_{i-1},c_i}$. Line 1 of Eqn. (10) assigns the value $-\infty$ if the codon assignment $c_{i-k},\ldots,c_{i-1},c_i$ is not *valid*. For all assignments which are *valid*, the equation (line 2) determines the CAI score by multiplying the optimal score up to position $i-1$ (guaranteed optimal by our assumption) with the fitness of the codon represented by c_i for amino acid α_i. Since every assignment to codon c_{i-k} is evaluated and the maximum is determined, then it must be the case that $P^i_{c_{i-k+1},\ldots,c_{i-1},c_i}$ correctly determines the score of the optimal *valid* codon design up to the i$^{\text{th}}$ codon position, having the codon assignment $c_{i-k+1},\ldots,c_{i-1},c_i$ for the last k codons, given that the maximum length of any motif is $3k$. □

Theorem 1. *$\widetilde{P^i_k}$ of Eqn. (10) correctly determines the score of an optimal* valid *codon design, with respect to CAI value, up to the i^{th} codon, given that the maximum length of any motif is $3k$.*

Proof. Lemma 1 guarantees that $P^i_{c_{i-k+1},\ldots,c_{i-1},c_i}$ correctly determines the score of the optimal *valid* codon design up to the i$^{\text{th}}$ codon, having the codon assignment $c_{i-k+1},\ldots,c_{i-1},c_i$ for the last k codons, given that the maximum length of any motif is $3k$. Therefore, if every possible assignment of the last k codons is evaluated, a maximum of 6^k possibilities, the score of an optimal *valid* codon design ending at position i can easily be determined.

First, consider that $\widetilde{F^i_k}$ correctly determines the minimum number of forbidden motifs possible, up to position i, by evaluating all possible assignments of that last k codons. Similarly, $\widetilde{D^i_k}$ evaluates the maximum number of desired motifs possible, by first ensuring that the minimum number of forbidden motifs criteria is satisfied. Finally, by evaluating all possible codon assignments of the last k codons, and determining the maximum score of all those which are *valid*, $\widetilde{P^i_k}$ must determine the optimal *valid* CAI score, up to position i. □

Under the assumption that the maximum length of any motif is constant, Theorem 2 proves that the overall time and space complexity is linear.

Theorem 2. *The dynamic programming algorithm for CAI optimization finds a valid nucleic acid sequence design for an amino acid sequence A in $O(|A|+h)$ time and $O(|A|+h)$ space, where h is the total length of forbidden and desired motifs and all motifs are of constant length.*

4 Empirical Results

Data set We use a filtered set of the 3,891 CDS (coding DNA sequence) regions of the GENECODE subset of the ENCODE dataset [15] (version hg17 NCBI build 35). This curated dataset comprises approximately 1% of the human genome and is representative of several its characteristics such as distribution of gene lengths and GC composition (54.31%). After filtering any sequences less than 75 bases in length, the remaining 3,157 CDS regions range in length from 75 to 8186 bases, averaging 173 bases with 267 bases standard deviation.

Codon frequencies In all cases, we use the codon frequencies of *Escherichia coli* as reported by the Codon Usage Database [http://www.kazusa.or.jp/codon].

Implementation and hardware All algorithms were implemented in C++ and compiled with g++ (GCC 4.1.0). Experiments were run on our reference Pentium IV 2.4 GHz processor machines, with 1GB main memory and 256 Kb of CPU cache, running SUSE Linux version 10.1.

4.1 Results

To evaluate the effectiveness and efficiency of our algorithm, a forbidden and desired motif set were constructed which could be considered typical in practice.

It is common for a gene synthesis experiment to use a single restriction enzyme. Furthermore, for reasons affecting gene expression, a common task is the removal of polyhomomeric regions (consecutive repeat region of identical nucleotides). Therefore, we have created a forbidden motif set containing ten elements including `GAGTC`, `GACTC`, `AAAA`, `TTTT`, `GGGG`, and `CCCC` where `GAGTC` is the motif for the *MlyI* restriction enzyme, `GACTC` is its reverse complement and the other motifs ensure no polyhomomeric regions greater than length three are permitted. The other four elements of the forbidden motif set (not shown) are immuno-inhibitory motifs originally used in the work of Satya *et. al* [13]. That work also used a desired motif set consisting entirely of thirty-three immuno-stimulatory motifs. We use this same desired motif set in our study.

Performance of the CAI optimization algorithm

motif sets	CAI value (std. dev.)	# forbidden (std. dev.)	# desired (std. dev.)
none (wild-types)	0.6477 (0.06)	9.2372 (16.24)	0.4869 (1.06)
forbidden	0.9161 (0.04)	0.1384 (0.45)	0 (0.00)
forbidden and desired	0.8280 (0.05)	0.1384 (0.45)	10.1324 (14.84)

Table 1. The mean values and standard deviations (averaged over 3,157 sequences) of CAI score, number of forbidden motifs, and number of desired motifs are shown for the original sequences (wild-types), the optimized sequences with forbidden motifs minimized and the optimized sequences with forbidden motifs minimized and then desired motifs maximized.

Results are shown for all 3,157 sequences in Figure 2. On the left side of the figure, the difference in optimal CAI value and the original CAI value of each sequence, when forbidden motifs are minimized, is plotted against sequence length. Desired motifs were not considered. For all sequences, the CAI value is improved compared with the original, with an average improvement of approximately 0.27. Shown on the right is the difference in CAI value for each sequence when the forbidden motifs are minimized and then the desired motifs are maximized. For this case, the average improvement of CAI value drops to 0.18, with only 12 sequences (0.4%) reporting a worse CAI value than the original. A summary of CAI statistics is presented in Table 1. In virtually all cases, forbidden motifs were eliminated entirely. Less than 2% of all sequences contained more than one forbidden motif after optimization, with only 0.6% containing more than two. On average 10 motifs were added to optimized sequences, when desired motifs were considered. These results demonstrate that it is possible to engineer motifs while still optimizing codon usage considerably. The runtime of the algorithm scales linearly with sequence length as would be expected. Considering desired motifs, in addition to forbidden motifs, increases run-time by a small constant

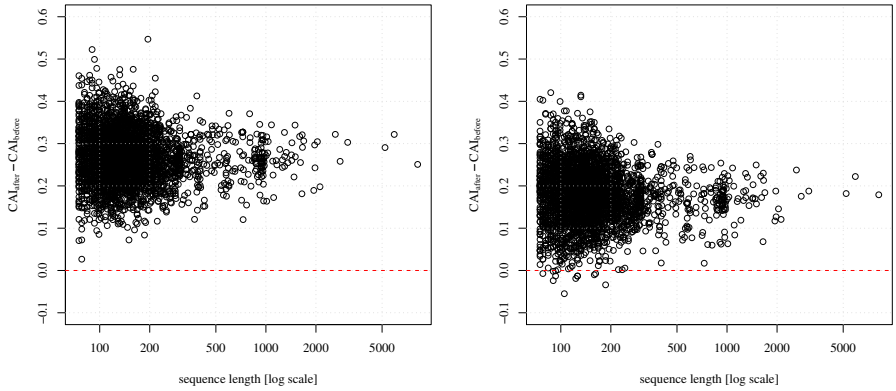

Fig. 2. Results are shown for the difference between the optimal CAI value and the original CAI value, plotted against sequence length, for each of the 3,157 sequences. On the left, results are shown when only the forbidden motif set is considered. The right side shows the results when both the forbidden and desired motif sets are considered.

factor, on average. In the worst case, the algorithm terminates in 0.43 CPU seconds for the longest sequence (8,141 bases).

5 Conclusions

In this work we have presented the first linear time and space algorithm for the problem of optimizing the codon adaptation index (CAI) value of a gene. The algorithm provides a guarantee that codon designs will be found which have a minimum number of forbidden motifs from some user defined set. The algorithm is also capable of adding desirable motifs, when applicable. A formal proof of correctness and time and space analysis was given. An extensive empirical analysis of the algorithm has shown it to be highly effective and efficient in practice. An efficient algorithm is a crucial first step towards designing genes while considering other important sequence features. For instance, designing genes with a guarantee that the resulting nucleic acid sequence does not form stable nucleic acid secondary structure is an interesting future direction, and one that may greatly effect translational efficiency.

Acknowledgments. The authors would like to thank the anonymous reviewers for their constructive suggestions to improve the presentation of this manuscript.

References

1. Aho, A.V.: Algorithms for finding patterns in strings, pp. 255–300. MIT Press, Cambridge, MA, USA (1990)
2. Fuglsang, A.: Codon optimizer: a freeware tool for codon optimization. Protein Expression and Purification 31(2), 247–249 (2003)
3. Gao, W., Rzewski, A., Sun, H., Robbins, P.D., Gambotto, A.: Upgene: Application of a web-based dna codon optimization algorithm. Biotechnology Progress 20(2), 443–448 (2004)
4. Grote, A., Hiller, K., Scheer, M., Münch, R., Nörtemann, B., Hempel, D.C., Jahn, D.: Jcat: a novel tool to adapt codon usage of a target gene to its potential expression host. Nucleic Acids Research 33(Web Server issue), 526–531 (2005)
5. Gusfield, D.: Algorithms on strings, trees, and sequences. Cambridge Press, New York, NY, USA (1997)
6. Gustafsson, C., Govindarajan, S., Minshull, J.: Codon bias and heterologous protein expression. Trends in Biotechnology 22(7), 346–353 (2004)
7. Holm, L.: Codon usage and gene expression. Nucleic Acids Research 14(7), 3075–3087 (1986)
8. Hoover, D.M., Lubkowski, J.: Dnaworks: an automated method for designing oligonucleotides for pcr-based gene synthesis. Nucleic Acids Research 30(10), e43 (2002)
9. Jayaraj, S., Reid, R., Santi, D.V.: Gems: an advanced software package for designing synthetic genes. Nucleic Acids Research 33(9), 3011–3016 (2005)
10. Kane, J.F.: Effects of rare codon clusters on high-level expression of heterologous proteins in escherichia coli. Current Opinion in Biotechnology 6(5), 494–500 (1995)
11. Lithwick, G., Margalit, H.: Hierarchy of sequence-dependent features associated with prokaryotic translation. Genome Research 13(12), 2665–2673 (2003)
12. Puigbo, P., Guzman, E., Romeu, A., Garcia-Vallve, S.: OPTIMIZER: a web server for optimizing the codon usage of DNA sequences. Nucleic Acids Research 35(suppl_2), W126–131 (2007)
13. Satya, R.V., Mukherjee, A., Ranga, U.: A pattern matching algorithm for codon optimization and cpg motif-engineering in dna expression vectors. In: CSB '03: Proceedings of the IEEE Computer Society Conference on Bioinformatics. pp. 294–305. IEEE Computer Society, Washington, DC, USA (2003)
14. Sharp, P.M., Li, W.H.: The codon adaptation index–a measure of directional synonymous codon usage bias, and its potential applications. Nucleic Acids Research 15(3), 1281–1295 (1987)
15. The ENCODE Consortium: The ENCODE (ENCyclopedia of DNA elements) project. Science 306(5696), 636–640 (2004)
16. Varenne, S., Lazdunski, C.: Effect of distribution of unfavourable codons on the maximum rate of gene expression by an heterologous organism. Journal of Theoretical Biology 120(1), 99–110 (1986)
17. Villalobos, A., Ness, J.E., Gustafsson, C., Minshull, J., Govindarajan, S.: Gene Designer: a synthetic biology tool for constructing artificial DNA segments. BMC Bioinformatics 7, 285 (2006)
18. Wu, G., Bashir-Bello, N., Freeland, S.J.: The synthetic gene designer: a flexible web platform to explore sequence manipulation for heterologous expression. Protein Expression and Purification 47(2), 441–445 (2006)