Bioinformatics, 36(13), 2020, 4012–4020
doi: 10.1093/bioinformatics/btaa248
Advance Access Publication Date: 20 April 2020
Original Paper

Gene expression

Codon optimization: a mathematical programing approach

Alper Şen[1,*], Kamyar Kargar[1], Esma Akgün[2] and Mustafa Ç. Pınar[1]

[1]Department of Industrial Engineering, Bilkent University, Ankara 06800, Turkey and [2]Department of Management Sciences, University of Waterloo, Waterloo, ON N2L 3G1, Canada

*To whom correspondence should be addressed.

Associate Editor: Bonnie Berger

Received on November 28, 2018; revised on November 25, 2019; editorial decision on April 7, 2020; accepted on April 13, 2020

Abstract

Motivation: Synthesizing proteins in heterologous hosts is an important tool in biotechnology. However, the genetic code is degenerate and the codon usage is biased in many organisms. Synonymous codon changes that are customized for each host organism may have a significant effect on the level of protein expression. This effect can be measured by using metrics, such as codon adaptation index, codon pair bias, relative codon bias and relative codon pair bias. Codon optimization is designing codons that improve one or more of these objectives. Currently available algorithms and software solutions either rely on heuristics without providing optimality guarantees or are very rigid in modeling different objective functions and restrictions.

Results: We develop an effective mixed integer linear programing (MILP) formulation, which considers multiple objectives. Our numerical study shows that this formulation can be effectively used to generate (Pareto) optimal codon designs even for very long amino acid sequences using a standard commercial solver. We also show that one can obtain designs in the efficient frontier in reasonable solution times and incorporate other complex objectives, such as mRNA secondary structures in codon design using MILP formulations.

Availability and implementation: http://alpersen.bilkent.edu.tr/codonoptimization/CodonOptimization.zip.

Contact: alpersen@bilkent.edu.tr

1 Introduction

A codon is a sequence of three nucleotides that encodes for a specific amino acid in the synthesis of a protein. There are 64 distinct codons, but only 20 amino acids, leading to the degeneracy of the genetic code. For instance, the amino acid *Leucine* can be encoded with six synonymous codons *CUU, CUC, CUA, CUG, UUA* and *UUG* whereas *Cysteine* can be encoded with two codons *UGU* and *UGC*. Overall, 2 of the 20 amino acids can be encoded with 1 codon, 9 with 2 codons, 1 with 3 codons, 5 with 4 codons and 3 with 6 codons leading to 61 essential codons. The remaining three codons are stop codons and are reserved for termination of protein formation. Codon degeneracy leads to many possible ways of encoding a protein, e.g. a typical 375-amino acid protein in humans can be potentially encoded by 10^{207} different codon sequences. All possible encodings and resulting sequences are not equally likely to be observed in nature; however, as some synonymous codons are more frequently used than others in encoding a particular amino acid in a particular organism. This phenomenon is called 'codon usage bias' or 'codon bias'. As an example, *Leucine* is encoded 39.5% of the time with codon *CUG* in *Homo sapiens*, whereas the same codon is used 11.1% of the same amino acid's encoding in *Saccharomyces cerevisiae* (Nakamura *et al.*, 2000).

Gene synthesis is now an important tool in many fields including production of bio-pharmaceuticals, diagnosis of diseases, vaccine development and gene therapy. Synthetic genes are inserted into the genetic material of various host organisms, such as bacteria and yeast to express and produce proteins. While researchers continue to identify new factors that affect the level of gene expression on host organisms, the effect of codon bias has long been known (Bennetzen and Hall, 1982; Gouy and Gautier, 1982). In fact, codon usage is shown to be the single most important factor in gene expression (Lithwick and Margalit, 2003). Using more frequently observed codons in the host organism instead of rarely observed ones increases the efficiency of the translation and the level of expression. Drastic—as much as 10^5-fold—improvements in expression levels are possible when right codons are used in the design of genes (Gustafsson *et al.*, 2004).

In order to measure how successful a particular design is in its use of codons that are more frequently observed in a host organism, Sharp and Li (1987) developed a metric called Codon Adaptation Index (CAI). This metric is based on what is called the fitness value of a codon for expressing an amino acid in a particular species. The fitness value of a codon is the ratio of its observed frequency to the observed frequency of the most frequent codon. For example, for *Cysteine*, the observed frequencies of codons *UGC* and *UGU* in

humans are 54.3% and 45.7%, respectively. This leads to fitness values $\tau(Cysteine, UGC) = 1$ and $\tau(Cysteine, UGU) = 0.457/0.543 = 0.842$. Formally, the fitness value of codon k in expressing amino acid i is given by $\tau(\gamma_i, \sigma_k) = \varphi^c_k / \max_{\ell \in K_i} \varphi^c_\ell$, where φ^c_ℓ is the observed frequency of codon ℓ in the species in consideration and K_i is the set of codons that can be used to express amino acid i. For a given amino acid sequence $\gamma = (\gamma_1, \gamma_2, \ldots, \gamma_N)$ of length N, the CAI of a codon sequence $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_N)$ is given by

$$\mathrm{CAI}(\gamma, \sigma) = \Big(\prod_{i=1}^{N} \tau(\gamma_i, \sigma_i) \Big)^{1/N}. \tag{1}$$

Clearly, CAI of a given codon sequence is between 0 and 1. Under no other restrictions, obtaining a CAI value of 1 is possible, which corresponds to using the most frequent codon for every amino acid.

Coleman *et al.* (2008) show that codon pair bias (CPB), i.e. the use of codon pairs that are more frequent in a host organism improves the level of gene expression. For example, while codons *GCC* and *UGC* are used 39.9% and 54.3% of the time in expressing amino acids *Alanine* and *Csyteine* in *H.sapiens*, respectively (leading to an expected frequency of 21.7% for the *GCC–UGC* codon pair), the observed frequency of codon pair *GCC–UGC* in *H.sapiens* is 36.9%. Buchan *et al.* (2006) show that codon pairing is biased in a diverse range of species and genomes. Coleman *et al.* (2008) suggest an index called CPB to measure the extent to which a codon sequence uses the frequently observed codon pairs. For a codon sequence $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_N)$ that encodes an amino acid sequence $\gamma = (\gamma_1, \gamma_2, \ldots, \gamma_N)$, this index is defined as

$$\mathrm{CPB}(\gamma, \sigma) = \frac{\sum_{i=1}^{N-1} \mathrm{CPS}(\gamma_i, \sigma_i, \gamma_{i+1}, \sigma_{i+1})}{(N-1)}, \tag{2}$$

where CPS $(\gamma_i, \sigma_k, \gamma_j, \sigma_\ell)$ is defined as codon pair score that compares the frequency of codon pair (σ_k,σ_ℓ) in encoding amino acid pair (γ_i,γ_j) relative to that expected by chance given the frequencies of each codon in the host organism. CPS is defined formally as follows:

$$\mathrm{CPS}(\gamma_i, \sigma_k, \gamma_j, \sigma_\ell) = \ln\left(\frac{\varphi^c_{k\ell} \varphi^a_i \varphi^a_j}{\varphi^a_{ij} \varphi^c_k \varphi^c_\ell} \right). \tag{3}$$

In this case, φ^a denotes the observed frequency of amino acids or pairs of amino acids in the species of interest. The CPS score for a given pair determines if the pair is over-represented (+) or under-represented (–) in the genome of a given species (Coleman *et al.*, 2008). Overall, a high value of CPB for a codon design means that the design uses more of the more frequent pairs and less of the less frequent pairs.

Gustafsson *et al.* (2004) argue that maximizing CAI (which corresponds to so-called 'one amino acid—one codon' approach, where one always attempts to encode an amino acid with the same codon) may often lead to translational errors due to imbalanced use of a subset of the tRNA. It is suggested that the objective should be to minimize the deviations from observed codon frequency of the host organism rather than to maximize the use of the most frequent codon. For this purpose, we define a metric called Relative Codon Bias, RCB(γ, σ) which measures how a codon sequence σ deviates from observed codon frequency when it is used to express amino acid sequence γ. This is formally defined as follows

$$\mathrm{RCB}(\gamma, \sigma) = \sum_{i \in A} \frac{\eta_i(\gamma)}{N} \sum_{j \in K_i} \frac{1}{|K_i|} \Big| \frac{\vartheta_j(\sigma)}{\eta_i(\gamma)} - \frac{\varphi^c_j}{\varphi^a_k} \Big|, \tag{4}$$

where K_i is the set of codons that can be used to express amino acid i (and $|K_i|$ is its cardinality), $\eta_i(\gamma)$ is the number of times amino acid i appears in the amino acid sequence γ and $\vartheta_j(\sigma)$ is the number of times codon j appears in the codon sequence σ. This metric is similar to one defined in Fox and Erill (2010) for measuring codon usage difference of a gene relative to a class of genes. Smaller values of RCB(γ, σ) correspond to codon designs that closely match observed codon usage in a given species.

Similar to RCB, one can define a metric, which measures how one particular codon design deviates from the observed frequencies of the codon pairs. Formally, this is called relative codon pair bias (RCPB) and defined as follows:

$$\mathrm{RCPB}(\gamma, \sigma) = \sum_{i,j \in A} \frac{\eta_{ij}(\gamma)}{N-1} \sum_{k \in K_i, \ell \in K_j} \frac{1}{|K_i||K_j|} \Big| \frac{\vartheta_{k\ell}(\sigma)}{\eta_{ij}(\gamma)} - \frac{\varphi^c_{k\ell}}{\varphi^a_{ij}} \Big|, \tag{5}$$

where $\eta_{ij}(\gamma)$ is the number of times amino acid pair ij appears in the amino acid sequence γ and $\vartheta_{k\ell}(\sigma)$ is the number of times the codon pair $k\ell$ appears in codon design σ.

The objective in codon optimization is to use synonymous codon changes in the gene such that one or more of the metrics mentioned above are optimized; eventually leading to an increase in protein production. In some cases, one also needs to ensure that certain forbidden motifs (nucleotide subsequences) are avoided and certain desired motifs are included. Given an extremely large number of possible codon designs, various software solutions are developed to support codon optimization over the years. Pioneering solutions in this area typically use a single objective (mainly CAI, requiring one to only substitute rare codons with codons that are most frequently observed in the host organism) and are reviewed in Villalobos *et al.* (2006). Clearly, the problem is a multi-objective one in nature, requiring more than one metric to be optimized simultaneously. Recent software solutions in this area, such as COOL (Chin *et al.*, 2014), D-Tailor (Guimaraes *et al.*, 2014), COStar (Liu *et al.*, 2014) and EuGene (Gaspar *et al.*, 2012) are able to handle multiple objectives and are reviewed in detail in Webster *et al.* (2017). A more comprehensive review of codon optimization algorithms and software solutions is provided in Gould *et al.* (2014). As also noted by the reviews, these solutions rely on heuristics and do not provide guarantees on optimality.

One exception in literature is a study by Condon and Thachuk (2012) who developed a dynamic programing algorithm that optimizes three objectives sequentially. In addition to this work, Papamichail *et al.* (2018) show that a codon design, which maximizes (or minimizes) CPB while ensuring that the individual codon frequencies (thus CAI) remain constant can be obtained using dynamic programing with a worst-case complexity of $O(N^{42})$, where N is the size of the sequence. Given that this is impractical for typical genes, Papamichail *et al.* (2018) resort to a simulated annealing heuristic. This article is the first to adopt a mathematical programing approach for the codon optimization problem. For the same problem that Papamichail *et al.* (2018) consider, our numerical results show that the mathematical approach we follow leads to significantly better sequences in terms of CPB in significantly shorter solution times. This study is also the first one that provides a mathematical guarantee of optimality for objectives involving RCB and RCPB.

We follow a mixed integer linear programing (MILP) approach for codon optimization. MILP is an optimization problem of the form $\min \mathbf{c}^T \mathbf{x}$ s.t. $\mathbf{A}\mathbf{x} \leq \mathbf{b}$ where $\mathbf{c}$ is a vector in $\mathbb{R}^{m+n}$, $\mathbf{b}$ is a vector in $\mathbb{R}^p$, $\mathbf{A}$ is a $p \times (m+n)$ matrix and decision variables $\mathbf{x} \in \mathbb{Z}^m \times \mathbb{R}^n$. While MILP is an NP-Hard problem, many scientific and commercial solvers are developed over the years and are in use for many successful real-life applications. In our case, we use integer (binary) variables to specify whether a particular codon is used to encode a given amino acid in the protein to be expressed and other decision variables to represent the metrics in terms of codon assignments. We use the four metrics CAI, CPB, RCB and RCPB as our objectives. In our first model, we use CAI and CPB in a bi-objective framework. In the second model, we use RCB and RCPB as our objectives. For both models, we can compute a set of solutions in the efficient frontier. We extend our formulation to consider secondary structures. Our results show that the mathematical programing approach we follow is robust; these and other various objectives and constraints can easily be incorporated in the formulations and truly optimal (or Pareto optimal) gene designs can be obtained for regular-sized proteins in acceptable solution times.

2 Materials and methods

We develop a MILP formulation for the codon optimization problem. The problem is a multi-objective optimization problem in nature as there are possibly multiple metrics that need to be considered. In this article, we consider the four objectives described in Section 1. Although more than two objectives can be easily handled, we present formulations for two bi-objective problems. We use the so-called ϵ-constraint approach, where one optimizes one objective function and the remaining objective is used as a constraint to create Pareto-optimal solutions.

Our first formulation (MaxCPBstCAI) maximizes CPB subject to CAI not falling below a specified value (α_{CAI}). We denote A to be the set of 20 amino acids, K to be the set of 61 codons. For each amino acid i, K_i represents the set of codons that amino acid i can be expressed with ($\cup_{i\in A} K_i = K$). In addition, the model requires the fitness value for each codon and CPS score for each codon pair for the host organism. Finally, the user needs to input α_{CAI}, the minimum value of CAI for the codon design. The model uses two types of decision variables. First, for all $i = 1,\ldots,N$ and $k \in K_{[i]}$ ($[i]$ stands for the specific amino acid in the ith place in the sequence), we define

$$x_{ik} = \begin{cases} 1, & \text{if } i\text{th amino acid is assigned to codon } k,\\ 0, & \text{otherwise.}\end{cases}$$

We also define, for all $i = 1,\ldots,N-1$, $j \in K_{[i]}$ and $k \in K_{[i+1]}$

$$z_{ik\ell} = \begin{cases} 1, & \text{if } i\text{th and } i+1\text{st amino acids are assigned to codons } k \text{ and } \ell\\ 0, & \text{otherwise.}\end{cases}$$

We are now ready to present our formulation.

(MaxCPBstCAI)

$$\max \quad \frac{\sum_{i=1}^{N-1}\sum_{j\in K_{[i]},k\in K_{[i+1]}} \text{CPS}([i],j,[i+1],k) z_{ijk}}{(N-1)} \tag{6}$$

$$\text{s.t.} \quad \sum_{k\in K_{[i]}} x_{ik} = 1,\ \text{i} = 1,\ldots,N, \tag{7}$$

$$z_{ik\ell} - \frac{1}{2}(x_{i,k} + x_{i+1,\ell}) \le 0,\ i = 1,\ldots,N-1,\ \forall k \in K_{[i]}, \ell \in K_{[i+1]}, \tag{8}$$

$$\sum_{i=1}^{N}\sum_{k\in K_{[i]}} \ln(\tau([i],k)) x_{ik} \ge N\ln(\alpha_{\text{CAI}}), \tag{9}$$

$$x_{ik} \in \{0,1\},\ i = 1,\ldots,N,\ \forall k \in K_{[i]}, \tag{10}$$

$$z_{ik\ell} \in \{0,1\},\ \forall i = 1,\ldots,N-1,\ \forall k \in K_{[i]},\ \forall \ell \in K_{[i+1]}. \tag{11}$$

The objective function (6) ensures that CPB is maximized. Constraint (7) ensures that each amino acid is assigned to exactly one codon. Constraint (8) ensures that the variable z_{ijk} can be set to one only if amino acid i is assigned to codon j and amino acid i+1 is assigned to codon k. Constraint (9) ensures that the CAI of the codon design does not fall below a specified input value and uses the fact that $\ln(\text{CAI}(\gamma,\sigma)) = \frac{1}{N}\sum_{i=1}^{N}\ln(\tau(\gamma_i,\sigma_i))$. Constraints (10) and (11) express that the decision variables are binary. The formulation (MaxCPBstCAI) has $O(N)$ binary decision variables and $O(N)$ constraints.

Our second model (MinRCPBstRCB) minimizes RCPB subject to RCB not exceeding a specified value. For a given amino acid sequence γ, the model requires the number of times each amino acid appears ($\eta_i(\gamma), i \in A$) and the number of times each amino acid pair appears ($\eta_{ij}(\gamma), i,j \in A$). In addition, the observed frequencies of amino acids ($\varphi_i^a, i \in A$), amino acid pairs ($\varphi_{ij}^a, i,j \in A$), codons ($\varphi_k^c, k \in K$) and codon pairs ($\varphi_{k\ell}^c, k,\ell \in K$) are required for the species at which the gene is to be expressed.

(MinRCPBstRCB)

$$\min \quad \sum_{i,j\in A}\frac{\eta_{ij}(\gamma)}{N-1}\sum_{k\in K_i,\ell\in K_j}\frac{1}{|K_i||K_j|}e_{k\ell}, \tag{12}$$

$$\text{s.t.} \quad \sum_{k\in K_{[i]}} x_{ik} = 1,\ i = 1,\ldots,N, \tag{13}$$

$$z_{ik\ell} - \frac{1}{2}(x_{i,k} + x_{i+1,\ell}) \le 0,\ i = 1,\ldots,N-1,\ \forall k \in K_{[i]}, \ell \in K_{[i+1]}, \tag{14}$$

$$e_{k\ell} - \frac{1}{\eta_{ij}(\gamma)}\sum_{h=1}^{N} z_{hk\ell} + \frac{\varphi_{k\ell}^c}{\varphi_{ij}^a} \ge 0,\ \forall i,j \in A,\ \forall k \in K_i, \forall \ell \in K_j, \tag{15}$$

$$e_{k\ell} + \frac{1}{\eta_{ij}(\gamma)}\sum_{h=1}^{N} z_{hk\ell} - \frac{\varphi_{k\ell}^c}{\varphi_{ij}^a} \ge 0,\ \forall i,j \in A,\ \forall k \in K_i, \forall \ell \in K_j, \tag{16}$$

$$\sum_{i\in A}\frac{\eta_i(\gamma)}{N}\sum_{k\in K_i}\frac{1}{|K_i|}d_k \le \alpha_{\text{RCB}}, \tag{17}$$

$$d_k - \frac{1}{\eta_i(\gamma)}\sum_{h=1}^{N} x_{hk} + \frac{\varphi_k^c}{\varphi_i^a} \ge 0,\ \forall i \in A, \forall k \in K_i, \tag{18}$$

$$d_k + \frac{1}{\eta_i(\gamma)}\sum_{h=1}^{N} x_{hk} - \frac{\varphi_k^c}{\varphi_i^a} \ge 0,\ \forall i \in A, \forall k \in K_i, \tag{19}$$

$$x_{ik} \in \{0,1\},\ i = 1,\ldots,N,\ \forall k \in K_{[i]}, \tag{20}$$

$$z_{ik\ell} \in \{0,1\},\ \forall i = 1,\ldots,N-1,\ \ \forall k \in K_{[i]},\ \forall \ell \in K_{[i+1]}, \tag{21}$$

$$d_k \ge 0,\ \forall k \in K, \tag{22}$$

$$e_{k\ell} \ge 0,\ \forall k,\ell \in K. \tag{23}$$

The decision variables in this formulation (in addition to x_{ik} and $z_{ik\ell}$) are d_k, which measures the deviation of codon k's frequency from its observed frequency and $e_{k\ell}$, which measures the deviation of codon pair $k\ell$'s frequency from its observed frequency. The objective function in (12) minimizes RCPB. Constraints (15) and (16) are used to linearize the absolute value function required for the deviation of frequency of codon pairs in the design from that of what is observed in the species. The constraint (17) ensures that the RCB does not exceed a specified value. The constraints (18) and (19) are used to linearize the absolute value function for the deviation of frequency of codons in the design from observed frequency. Finally, constraints (22) and (23) state that the decision variables used for frequency deviations are non-negative continuous variables. The formulation (MinRCPBstRCB) has $O(N)$ binary variables and a fixed number ($|K|^2 + |K|$) of continuous decision variables and $O(N)$ constraints.

The mathematical programing formulation developed above is modeled using Gurobi Python interface (http://www.gurobi.com). The Python code reads as input (i) amino acid sequence for the protein to be expressed in the host organism (ii) the objectives to be considered (iii) observed frequencies of amino acids and codons (and their pairs) in the desired host organism and (iv) user-specified values for the other objectives.

The first step in the program is to add decision variables for the chosen model. The program then adds the objective provided in (6) or (12). The program then adds constraints. The program then solves the model chosen and outputs the final codon design in a text file, along with the value of the objective function(s) that are chosen.

3 Results

In order to evaluate the effectiveness of our MILP formulations, we used protein sequences from the UniProt Database (The UniProt Consortium, 2018). The Uniprot Database had information for 559 634 proteins at the time of accession. The number of amino acids in a protein in this database range from minimum 2 to a maximum 35 213 with a median of 294. We sampled two proteins from each fifth percentile (in the number of amino acids) leading to a total number of 40 proteins to be analyzed. Codon pair frequencies for only *H.sapiens* were available in Coleman *et al.* (2008), therefore, we did the analysis for *H.sapiens*. The codon frequencies are obtained from Codon Usage Database (Nakamura *et al.*, 2000).

MILP formulation is developed using Gurobi's Python (version 3.7.2) interface and the models are solved using Gurobi solver version 8.1.1. All problems are run on a computer with a 2.3 GHz processor and 2 GB main memory, running on Windows version 10. All problems are solved to optimality with the default settings of Gurobi. The exception is the parameter MIPGap (set to 0.001), which is the gap between lower and upper objective bounds, below which the solver will conclude that it found the optimal solution and terminate.

In our first model, we maximize CPB subject to CAI not falling below a specified value using the model (MaxCPBstCAI). We use 10 different values for CAI in the range (0.55, 1.00) with 0.05 increments. For each CAI value, a separate model is run. An example of such analysis is shown for a 367 amino acid protein with ID RL10PROM0 in Figure 1. CPB can be as low as −0.046 when CAI is 1.0 (when one uses the most frequent codon for every amino acid), and as high as 0.386 when CAI is allowed to be 0.55. The other points in the plot correspond to different Pareto-optimal solutions found by the model creating the efficient frontier for CPB and CAI.

The solution times for all of our experiments for CPB versus CAI are shown in Figure 2 (model building times are excluded from these times. Model building times are 1.20 s for the smallest protein and 18.94 for the largest protein. We note that for the same protein with different CAI values, the model has to be built only once). Each point in the graph corresponds to a single run of the model (MaxCPBstCAI) for one protein and one CAI value. Overall, solution time increases as the number of amino acids in the sequence increases, but all solution times are below 6 s. We also ran our model for the largest protein in the UniProt Database which has 35 213 amino acids. The solver was able to generate an optimal solution (for a given CAI value) in roughly 5800 s. For these very rare extremely large proteins, one can consider splitting the sequence into smaller pieces and running separate optimization models to generate near-optimal solutions in more reasonable times.

We also compare our results with the simulated annealing approach proposed by Papamichail *et al.* (2018). We use the Codon Context Evaluation Tool (CCTool) developed by Papamichail *et al.* (2018), which is available at http://algo.tcnj.edu/cctool. This tool receives a nucleotide sequence and uses a simulated annealing algorithm to maximize the CPB subject to CAI not falling below CAI of the original sequence. In order to find the input sequences, we run the model (MaxCPBstCAI) but this time with a minimization objective for 9 different values of α_{CAI} (0.55, 0.60, …, 0.95) and for 40 proteins that we sample from the UniProt Database, resulting in 360 sequences. We set the number of iterations to 500 000 (the default was 5000) and let the tool maximizes CPB for each sequence. We record the objective function values and solution times and compare them with the results we obtain using our model.

For *all* input sequences, the sequence obtained by CCTool has a strictly smaller CPB value and the solution time was longer than those obtained using our model. CPB value obtained by CCTool was, on the average, 37.46% smaller than CPB value obtained by our model (minimum 14.50%, maximum 128.70%). The solution time using CCTool was, on the average, 4021.19% longer than the solution time of our model (minimum 223.28%, maximum 23 275.00%). All comparative results are reported in Figure 3. Overall, our model performs significantly better than the CCTool, both in terms of solution time and solution quality (objective value). We finally note that using smaller number of iterations (5000 or 50 000) in CCTool may decrease solution times. However, our

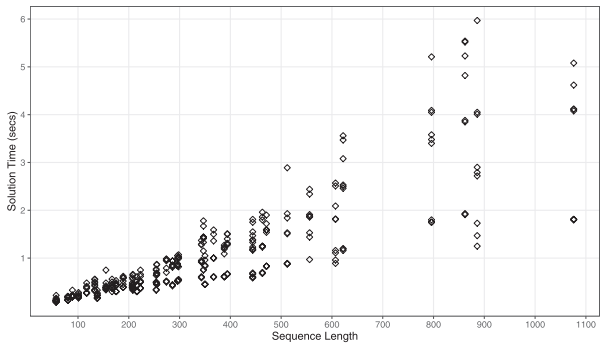

Fig. 2. Solution times (s) for CPB and CAI: solution times are for one run of the model (MaxCPBstCAI) and exclude model building times

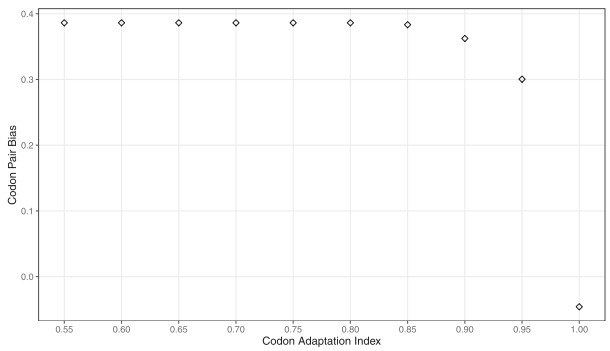

Fig. 1. An efficient frontier of CPB and CAI—Protein RL10PROM0: each point in the plot is obtained running the model (MaxCPBstCAI) once and corresponds to a codon design, which maximizes CPB subject to CAI not falling below a specified value

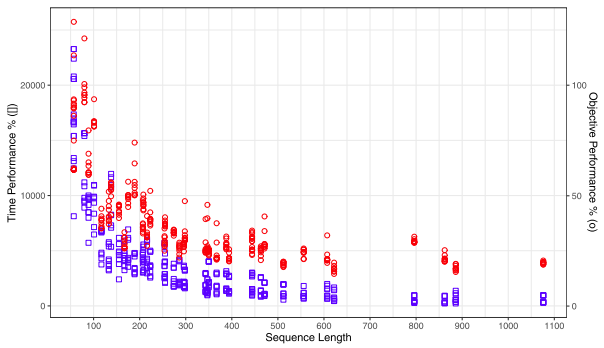

Fig. 3. Comparison of our model with CCTool: the primary *y*-axis and squares represent the performance gap of the CCTool relative to our model [100 × (CCTool-model)/model] in terms of the solution time; the secondary *y*-axis and circles represent the performance gap of the CCTool relative to our model in terms of the objective (CPB) value

approach still outperforms CCTool in solution time and the performance gap in terms of solution quality increases substantially, especially for large sequences.

In our second model, we minimize RCPB subject to RCB not exceeding a specified value. We use 10 different values for RCB in the range (0.055–0.100) with 0.005 increments. For each value of RCB, a separate model (MinRCPBstRCB) is run. An example of our analysis with protein RL10PROM0 is shown in Figure 4. For RCB not exceeding 0.10, one can find a codon design for which RCPB is roughly 0.1332. When RCB is not allowed to be above 0.055, the best codon design has a RCPB around 0.1358.

The solution times for all of our experiments for RCPB versus RCB is shown in Figure 5 where solution time is in logarithmic scale (these times do not include model building times. Model building times are 1.67 s for the smallest protein and 22.02 for the largest protein. We note that for the same protein with different RCB values, the model has to be built only once). Each point corresponds to a single run of the model (MinRCPBstRCB) for one protein and one RCB value. A total of 337 out of 360 instances can be solved within the time limit of 3600 s. For the remaining 23 instances, the sequence found at the end of 3600 s is guaranteed to be within 0.56% of the true optimal, on the average. Solution times clearly increase as the sequences get longer. We also note that for six instances, there is no feasible solution, meaning that one cannot find a feasible sequence with an RCB smaller than the input value. Again, we ran our model for the largest protein with 35 213 amino acids. The solver was not able to generate an optimal solution within 4 h. As mentioned above, we believe that for such very large proteins, a divide and conquer approach may be used without sacrificing much from optimality.

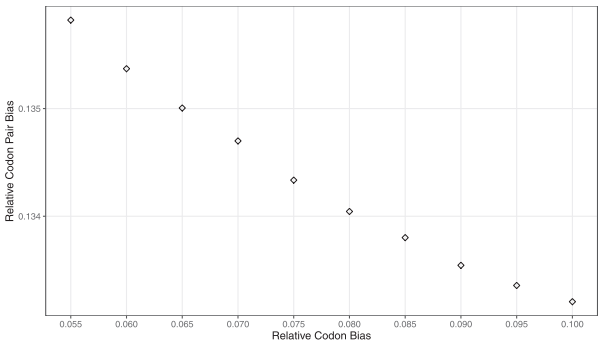

Fig. 4. An efficient frontier of relative CPB and RCB YCB—Protein RL10PROM0: each point in the plot is obtained running the model (MinRCPBstRCB) once and corresponds to a codon design, which minimizes RCPB subject to RCB not being above a specified value

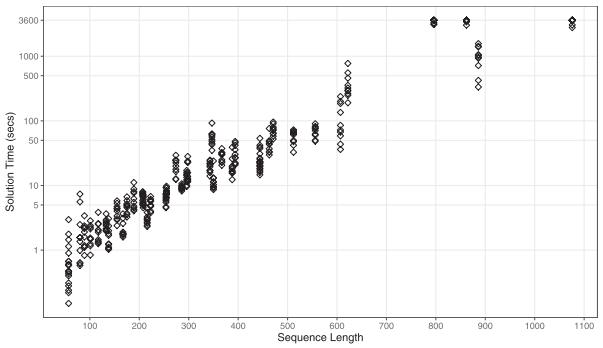

Fig. 5. Solution times for relative CPB and RCB: solution times are for one run of the model (MinRCPBstRCB) and exclude model building times

3.1 Codon optimization considering secondary structures

Many RNAs are known to fold in on themselves to be thermodynamically more stable. The particular folding pattern is described as a secondary structure, which is defined as a set of hydrogen-bonding base pairs (such as Watson–Crick pairs, *A–U* and *C–G*). Secondary structures in messenger RNA (mRNA) are known to have an effect on gene expression and protein production (Kudla *et al.*, 2009). A recent extensive design-of-experiments study shows that mRNA secondary structures have the biggest effect on translation efficiency and less stable structures (i.e. less folding), especially around the start codon, increases translation efficiency significantly (Cambray *et al.*, 2018). A number of gene design tools provide functionality regarding mRNA secondary structures (Gould *et al.*, 2014). mRNA optimizer (Gaspar *et al.*, 2013) uses a simulated annealing heuristic and a pseudo-energy assessor to predict the energy level of a given design. Visual Gene Developer (Jung and McDonald, 2011) asks user to specify a range of energy levels for a part of the gene and modifies the gene in an *ad-hoc* manner until predicted energy level falls inside this range. D-Tailor (Guimaraes *et al.*, 2014) only allows one to see the predicted secondary structure of a given sequence designed by the tool. As such, current tools require one to generate a set of codon designs and their mRNA secondary structures are predicted approximately or using folding packages, such as Mfold (Zuker, 2003). This may be time consuming especially when there are a large number of candidate sequences as is the case in codon optimization. Energy level or the stability of the secondary structure is not formally posed as a formal putative objective in codon optimization in the earlier literature, to the best of our knowledge.

In this section, we formulate the simultaneous optimization of codon sequence and mRNA secondary structures problem using a bi-level MILP approach. We note that, integer programing formulations are previously used to predict mRNA secondary structures (Poolsap *et al.*, 2009; Sato *et al.*, 2011) given a fixed nucleotide (therefore, a fixed codon) sequence. We allow the possibility of reasonably general secondary structures with simple pseudoknots in our formulation.

A simple pseudoknot is depicted in Figure 6. A simple pseudoknot needs to satisfy the following set of conditions: (i) each nucleotide can be paired to at most one other nucleotide (and these pairings should be one of the allowed pairings, e.g. *A–U*, *C–G*), (ii) a base pair from above (below) the sequence cannot cross another base pair above (below) the sequence, (iii) a nucleotide cannot be paired with another nucleotide within a distance of τ, (iv) the beginning of all base pairs above the sequence should be before the beginning of any base pair below the sequence and (v) the end of all base pairs above the sequence should be before the end of any base pair below the sequence. Given a nucleotide sequence, it is assumed that the secondary structure that will be formed is the one that gives the minimum free energy. The energy function depends on the adjacent base pairs (or stacking pairs), i.e. nucleotides k and ℓ paired through letters m and n and nucleotides $k+1$ and $\ell-1$ paired through letters o and p lead to an energy level μ^{mnop}. It is further assumed that stacking pairs below the sequence are penalized with a weight $\rho \leq 1$ (Rivas and Eddy, 1999). Any loop region in a pseudoknot can potentially have its own secondary structure; such structures are called recursive pseudoknots. Akutsu (2000) develops a dynamic programing algorithm to find the minimum energy folding for recursive pseudoknots. The complexity of this algorithm is $O(n^5)$, where n is the number of nucleotides in the sequence.

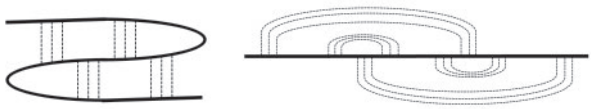

Fig. 6. Simple pseudoknot (and its linear organization on the right). Dashed lines correspond to base pairs

The use of such algorithms is not viable, especially when one has flexibility in codon choice, given the large number of nucleotides in typical genes. We finally note that, finding the minimum energy folding for structures more general than recursive pseudoknots is NP-Hard (Akutsu, 2000).

Part of our MILP formulation has similarities with the secondary structure prediction formulation in Poolsap *et al.* (2009). However, our formulation handles the pseudoknot-specific conditions [conditions (iv) and (v) above] in a more compact manner, has the flexibility to choose any of the synonymous codons for each amino acid and creates the association between codon sequences and nucleotide sequences. We now describe our formulation. Let x_{ij} be a binary variable which takes on the value one if the amino acid in the ith sequence is assigned to codon j and zero otherwise. Let y_{kn} be a binary variable, which has value one if the kth nucleotide in the sequence is assigned to letter n and zero otherwise. Let $\bar{u}_{k\ell}$ be a binary variable, which has value one if the kth nucleotide is paired with ℓth nucleotide from above the sequence and zero otherwise. $\underline{u}_{k\ell}$ is similarly defined for pairing from below the sequence. The binary variables $\bar{u}^{mn}_{k\ell}$ and $\underline{u}^{mn}_{k\ell}$ specify whether pairing for kth and ℓth nucleotides is through letters m and n. Finally the binary decision variable $\bar{w}^{mnop}_{k\ell}$ specifies whether nucleotides k and ℓ are paired through letters m and n while at the same time, nucleotides $k+1$ and $\ell-1$ are paired through letters o and p forming a stacking pair of the type $omnp$ above the sequence. The variable $\underline{w}^{mnop}_{k\ell}$ is defined similarly for stacking pairs below the sequence.

Define $\theta(k) = k - 3\lfloor (k-1)/3 \rfloor$. This function together with $\lceil k/3 \rceil$ is used to create a correspondence between the codons and letters in the sequence. For example, the 13th letter in the sequence is defined as the first letter ($\theta(13) = 1$) of the fifth assigned codon ($\lceil 13/3 \rceil = 5$) in the sequence. Let λ_{jk} defines the kth nucleotide in codon j. Our bi-level MILP formulation (MaxFECOstCAI) is as follows:

(MaxFECOstCAI)

$$\max\ h(\boldsymbol{x}, \boldsymbol{y}), \tag{24}$$

$$\text{s.t.} \quad \sum_{j \in K_{[i]}} x_{ij} = 1, \tag{25}$$

$$\sum_{i=1}^{N} \sum_{j \in K_{[i]}} \ln(\tau([i], j)) x_{ij} \geq N \ln(\alpha_{\text{CAI}}), \tag{26}$$

$$y_{kn} - \sum_{j \in K_{[\lceil k/3 \rceil]}} x_{\lceil k/3 \rceil, j} 1_{\{\lambda_{j,\theta(k)} = n\}} = 0, \tag{27}$$

$$x_{ij}, y_{kn} \in \{0, 1\}, \tag{28}$$

$$\text{where } h(\mathbf{x}, \mathbf{y}) = \min \sum_{k,\ell} \sum_{mn,op} (\mu^{mnop} \bar{w}^{mnop}_{k\ell} + \rho \mu^{mnop} \underline{w}^{mnop}_{k\ell}), \tag{29}$$

$$\text{s.t.} \quad \bar{u}^{mn}_{k\ell} - \frac{1}{2}(y_{km} + y_{ln}) \leq 0, \tag{30}$$

$$\underline{u}^{mn}_{k\ell} - \frac{1}{2}(y_{km} + y_{\ell n}) \leq 0, \tag{31}$$

$$\bar{u}_{k\ell} - \sum_{mn} \bar{u}^{mn}_{k\ell} = 0, \tag{32}$$

$$\underline{u}_{k\ell} - \sum_{mn} \underline{u}^{mn}_{k\ell} = 0, \tag{33}$$

$$\sum_{\ell} \bar{u}_{k\ell} + \sum_{\ell} \underline{u}_{k\ell} + \sum_{\ell'} \bar{u}_{\ell' k} + \sum_{\ell'} \underline{u}_{\ell' k} \leq 1, \tag{34}$$

$$\sum_{k' < k} \bar{u}_{k'\ell} + \sum_{\ell' > \ell} \bar{u}_{k\ell'} \leq 1,\ k < \ell, \tag{35}$$

$$\sum_{k' < k} \underline{u}_{k'\ell} + \sum_{\ell' > \ell} \underline{u}_{k\ell'} \leq 1,\ k < \ell, \tag{36}$$

$$\sum_{h} \bar{u}_{kh} + \sum_{h} \underline{u}_{\ell h} \leq 1,\ \ell < k, \tag{37}$$

$$\sum_{h} \bar{u}_{hk} + \sum_{h} \underline{u}_{h\ell} \leq 1,\ \ell < k, \tag{38}$$

$$\sum_{\ell \leq k-\tau} \bar{u}_{k\ell} + \sum_{\ell \leq k-\tau} \underline{u}_{k\ell} = 0, \tag{39}$$

$$\bar{w}^{mnop}_{k\ell} - \frac{1}{2}(\bar{u}^{mn}_{k\ell} + \bar{u}^{op}_{k+1,\ell-1}) \leq 0, \tag{40}$$

$$\underline{w}^{mnop}_{k\ell} - \frac{1}{2}(\underline{u}^{mn}_{k\ell} + \underline{u}^{op}_{k+1,\ell-1}) \leq 0, \tag{41}$$

$$\bar{u}_{k\ell}, \underline{u}_{k\ell}, \bar{u}^{mn}_{k\ell}, \underline{u}^{mn}_{k\ell}, \bar{w}^{mnop}_{k\ell}, \underline{w}^{mnop}_{k\ell} \in \{0, 1\}. \tag{42}$$

The first level of (MaxFECOstCAI) involves determining codon designs that will minimize the folding in the secondary structures subject to constraint (lower bound) on the CAI value. The decision variables in this level are $\boldsymbol{x} = (x_{ij})$ and $\boldsymbol{y} = (y_{kn})$. Objective (24) maximizes the weighted sum of free energies associated with stacking pairs in the sequence defined by $h(\boldsymbol{x}, \boldsymbol{y})$ and determined in the second level of (MaxFECOstCAI). Constraint (25) ensures that each amino acid is assigned to one codon that it can be expressed with. Constraint (26) ensures that the CAI value of the codon assignment does not fall below the input value α_{CAI}. Constraint (27) converts the codon assignments to nucleotide assignments in the sequence. For example, if the fifth amino acid in the sequence (*Cysteine*) is assigned the codon *UGC*, this constraint sets the 13th, 14th and 15th nucleotides to letters *U*, *G* and *C*, respectively.

The second level of (MaxFECOstCAI) involves predicting the folding structure given the codon assignments in the first level. Objective (29) minimizes the weighted sum of free energies associated with stacking pairs in the sequence. Constraint (30) allows a pairing of nucleotides k and ℓ through letters m and n above the sequence only if kth nucleotide is letter m and ℓth nucleotide is letter n. Constraint (31) is used similarly for pairing below the sequence. Constraints (32) and (33) specify whether any two nucleotides are paired above and below the sequence. Constraint (34) ensures that each nucleotide is paired with at most one nucleotide, before or after, above or below the sequence. Constraint (35) ensures that base pairs above the sequence do not cross each other. Constraint (36) ensures that base pairs below the sequence do not cross each other. Constraints (37) and (38) ensure that the base pairings do not violate the simple pseudoknot structure. Constraint (39) ensures that bases that are very close to each other are not paired together. Constraint (40) specifies whether base pairs (k, ℓ) and $(k+1, \ell-1)$ are stacked together with letters (m, n) and (o, p) above the sequence. Constraint (41) is defined similarly for stacking pairs below the sequence. Constraint (42) ensures that all decision variables are binary. All constraints are defined for all possible values of the free indices, unless stated otherwise.

The formulation (MaxFECOstCAI) is flexible enough to incorporate other objectives and constraints. For example, one can include constraints such that certain nucleotide subsequences are avoided (*forbidden motifs*) or used (*desired motifs*) to the extent possible. For example, if a motif $(m_1, m_2, \ldots, m_q)$ needs to be avoided altogether, one can incorporate the constraint:

$$\sum_{\ell=1}^{q} y_{k+\ell, m_\ell} \leq q - 1,\ k = 0, \ldots, N - q. \tag{43}$$

One can also count the number of times such motifs are used by introducing additional decision variables and use them to state constraints regarding their total count. Finally, it is easy to revise the formulation to consider a portion of the sequence if one is interested in folding energy only in that particular part of the sequence.

Solving bi-level programs in general is very difficult (DeNegre and Ralphs, 2009) primarily because the objective or the constraints in the first level ($h(\boldsymbol{x},\boldsymbol{y})$ in our case) are implicitly defined by the second-level optimization problem. Therefore, we utilize an alternative approximate approach. It is well-known that free energy parameters for structures that use *A–U* nucleotide pairs instead of *C–G* nucleotide pairs are less negative leading to less stable secondary structures. Therefore, if the codon assignments are determined such that the resulting nucleotide sequence is *A–U* rich, one would expect that the resulting secondary structures are less stable and more efficient in translation. In fact, the negative correlation between *A–U* content and folding has been shown in earlier studies (Bentele *et al.*, 2013; Seffens and Digby, 1999). In light of these, we set the objective of the first level problem as maximizing the total number of *A* and *U* nucleotides.

(MaxAUstCAI)

$$\max \sum_{k} y_{kA} + (1+\delta)\sum_{k} y_{kU}, \tag{44}$$

$$\text{s.t.} \quad \sum_{j\in K_{[i]}} x_{ij} = 1, \tag{45}$$

$$\sum_{i=1}^{N}\sum_{j\in K_{[i]}} \ln(\tau([i],j))x_{ij} \geq N\ \ln(\alpha_{\text{CAI}}), \tag{46}$$

$$y_{kn} - \sum_{j\in K_{[\lceil k/3\rceil]}} x_{\lceil k/3\rceil,j} 1_{\{\lambda_{j,\theta(k)}=n\}} = 0, \tag{47}$$

$$x_{ij}, y_{kn} \in \{0,1\}. \tag{48}$$

We use a different weight for the number of *U* nucleotides in the objective function (44) through the parameter $\delta \neq 0$. This is to create an imbalance between the number of *A* and *U* nucleotides and to further decrease the possibility of secondary structure formation. The resulting sequence obtained from the model (MaxAUstCAI) is then input into the model given by (29)–(42), now to predict the secondary structures and resulting energy levels.

We randomly select 12 smaller proteins from The UniProt Database in order to test the effectiveness of our formulation. We use the Watson–Crick pairs (*A–U* and *C–G*), but ignore the Wobble pairs (*U–G*) in our analysis. We use the stacking energy parameters provided in Poolsap *et al.* (2009). For each protein, for 10 different values of α_{CAI} $(0.55, 0.60, \ldots, 0.95, 1.0)$, we run the model (MaxAUstCAI) to obtain codon sequences. In doing this, we use fitness values of *Escherichia coli*. Resulting codon sequences are fed in to models (29)–(42) to predict secondary structures for three different values of ρ: 1, 0.8 and 0.6. We use a value of $\delta = 0.001$ in the objective function of (MaxAUstCAI). A total of 360 problems are solved. The software and hardware specifications are the same as before. The total solution times, i.e. solution time of (MaxAUstCAI) plus the solution time of models (29)–(42), are provided in Figure 7 where the *y*-axis is in logarithmic scale (model building times are excluded from these times. Combined model building times are 1.94 s for the smallest protein and 58.02 s for the largest protein).

We have the following observations from the results of our tests. For genes of small size (lower left part of Fig. 7) the solution times are reasonable, usually not exceeding 1 s. Due to the increased number of constraints and binary variables, the solution effort requires longer time as the number of amino acids increases. Most of the solution effort is spent for energy prediction performed using models (29)–(42) with the sequence obtained from the model (MaxAUstCAI). The time spent in solving the model (MaxAUStCAI) for all proteins used in the test is inferior to 0.1 s for every value of CAI we have chosen to keep fixed. In other words, codon sequences that are on the efficient frontier can be determined very rapidly.

An example is provided in Figure 8 for protein PSBXCYAPA, which has 39 amino acids. The horizontal axis is CAI. On the primary vertical axis, we have *AU*-Content which is presented as the number of nucleotides, which are letter *A* or *U* (out of 117 nucleotides). In the secondary vertical axis, we have free energy of the predicted secondary structure of the codon sequence in kcal/mol. As one can see, as minimum CAI level is allowed to be lower, the resulting codon sequences can have more *A* and *U* nucleotides and are less stable. For example, when CAI=1.0, *AU*-Content=62 and folding energy $=-48.4$ kcal/mol whereas when CAI=0.9, *AU*-Content $=78$ and folding energy $=-37.1$ kcal/mol.

The association between the energy levels and *AU*-Content is also observed for the other proteins that we studied. When *AU*-Content is presented as a fraction of the total number of nucleotides in a given sequence and free energy is presented as a fraction of the free energy when CAI=1 (i.e. both metrics are normalized to one), we found that the Pearson correlation coefficient between these two metrics is found to be -0.663 for $\rho=1$ for 120 different amino acid sequences (12 proteins and 10 different CAI values).

The summary results for all 12 proteins are presented in Table 1 for $\rho=1$, where we report the normalized values of *AU*-Content and free energy as explained above. Our results show that minimum free energy can be increased (can be made less negative) significantly when codons can be optimized. Not using the most frequent codons and allowing CAI to be at 0.95, e.g. may increase the free energy of

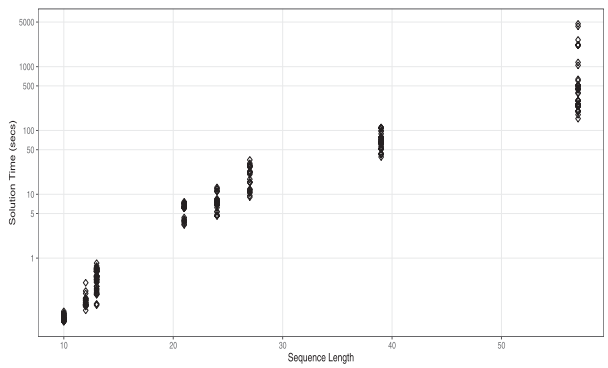

Fig. 7. Solution times for codon optimization considering secondary structures: solution times represent the sum of the solution time of the model (MaxFECOstCAI) and time of the prediction obtained through models (29)–(42) and exclude model building times

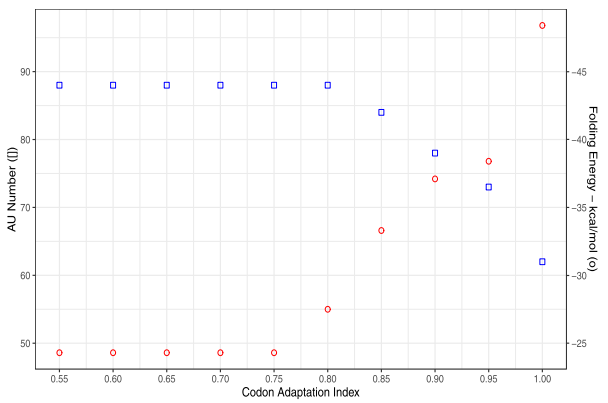

Fig. 8. Efficient frontier of CAI and folding energy/*AU*-content for protein PSBXCYAPA, $\rho=1$: the primary *y*-axis and squares represent the *AU*-content in the number of *A* and *U* nucleotides; the secondary *y*-axis and circles represent folding energy in kcal/mol

Table 1. Summary results for codon optimization considering secondary structures - CAI vs. averages of normalized values of *AU*-Content and free energy

CAI	0.70	0.75	0.80	0.85	0.90	0.95	1.00
AU-Content	0.733	0.733	0.667	0.633	0.567	0.500	0.400
Energy level	0.653	0.655	0.665	0.696	0.699	0.821	1.000

the secondary structures by 17.9% (note that all minimum free energies are negative) making them significantly less stable and allowing more efficient translation.

4 Discussion

Methodical gene optimization is considered to be impractical as it is intractable to consider all possible number of gene sequences for an average size protein (Welch *et al.*, 2009). Therefore, most codon optimization solutions developed in the past use the word 'optimization' vaguely. In the pioneering solutions, codon optimization simply refers to replacing rare codons with frequently used ones in a host organism, essentially following a 'one amino acid—one codon' approach. Examples include Codon Optimizer (Fuglsang, 2003), UpGene (Gao *et al.*, 2004) and JCat (Grote *et al.*, 2005). As this approach may lead to an imbalanced use of tRNA, a number of solutions, such as DNAWorks (Hoover and Lubkowski, 2002), GeneDesigner (Villalobos *et al.*, 2006) and OPTIMIZER (Puigbo *et al.*, 2007) also provide Monte Carlo algorithms that randomly select the codons based on the observed frequencies of codons in the host organism.

More recent solutions consider criteria other than codon bias, such as codon context bias and motif avoidance and provide multi-objective functionality. These solutions are EuGene (Gaspar *et al.*, 2012), COOL (Chin *et al.*, 2014), D-Tailor (Guimaraes *et al.*, 2014), COStar (Liu *et al.*, 2014) and a study by Gonzalez-Sanchez *et al.* (2019). However, all of these solutions use heuristics and thus do not provide a mathematical guarantee of obtaining an optimal solution (or Pareto-optimal solutions). COOL (Chin *et al.*, 2014) and D-Tailor (Guimaraes *et al.*, 2014) use genetic algorithms, EuGene (Gaspar *et al.*, 2012) uses a simulated annealing heuristic along with a genetic algorithm, COStar (Liu *et al.*, 2014) uses a D-star Lite-based dynamic search algorithm and Gonzalez-Sanchez *et al.* (2019) use an artificial bee colony algorithm.

To our knowledge, there are only four articles in the literature that provide a mathematical guarantee of obtaining an optimal solution for the gene design problems that are considered. Three of these articles (Condon and Thachuk, 2012; Satya *et al.*, 2003; Skiena, 2001) consider CAI along with the use of desired or forbidden motifs as objectives and provide polynomial-time algorithms. A study by Papamichail *et al.* (2018) considers codon context bias and provides a guarantee of optimality. When codon context bias (CPB) is considered alone, the authors develop an $O(N)$ algorithm. The authors show that when the codon bias is fixed (i.e. the frequencies of the codons used are to remain constant), the problem reduces to a version of the traveling salesman problem and can be solved with a time complexity of $O(N^{41})$. As the time requirements are not practical for even moderately sized protein sequences, the authors suggest a simple branch and bound algorithm. This algorithm does not scale either and the authors resort to a simulated annealing heuristic.

This article is the first to study the synthetic gene design problem using a mathematical programing approach. We show that various criteria, such as codon bias, CPB and RCB, can be easily modeled using this approach in a multi-objective framework. The mathematical programing approach allows one to compute a set of Pareto-optimal solutions. We show that one can obtain gene designs with a mathematical guarantee of (Pareto) optimality using this approach for real-size proteins in reasonable solution times. Our numerical results show that our model (maxCPBstCAI) can solve the same problem attacked by Papamichail *et al.* (2018) for very large proteins, with a guarantee of optimality, within 6 s. For all sequences that we analyzed, our approach leads to significantly shorter solution times and better sequences in terms of CPB. This article is also the first to explicitly consider secondary structure formation as an objective for codon optimization in an analytical formulation. Focusing only on critical parts of the sequence for secondary structure formation will make this approach viable for proteins that are larger than what we consider here.

The significance of our work derives also from the fact that general-purpose optimization software (such as Gurobi used in this work) can be used for solving mathematical programing problems (namely, MILP problems) obviating the need to develop customized heuristics or optimization algorithms (and modify them as new criteria are found to be important in gene expression). These software solutions are continually being developed further and are accessible to academic users gratis, as was the case in this article.

Financial Support: none declared.

Conflict of Interest: none declared.

References

Akutsu,T. (2000) Dynamic programming algorithms for RNA secondary structure prediction with pseudoknots. *Discrete Appl. Math.*, **104**, 45–62.

Bennetzen,J.L. and Hall,B.D. (1982) Codon selection in yeast. *J. Biol. Chem.*, **257**, 3026–3031.

Bentele,K. *et al.* (2013) Efficient translation initiation dictates codon usage at gene start. *Mol. Syst. Biol.*, **9**, 675.

Buchan,J.R. *et al.* (2006) tRNA properties help shape codon pair preferences in open reading frames. *Nucleic Acids Res.*, **34**, 1015–1027.

Cambray,G. *et al.* (2018) Evaluation of 244,000 synthetic sequences reveals design principles to optimize translation in *Escherichia coli*. *Nat. Biotechnol.*, **36**, 1005–1015.

Chin,J.X. *et al.* (2014) Codon Optimization OnLine (COOL): a web based multi-objective optimization platform for synthetic gene design. *Bionformatics*, **30**, 2210–2212.

Coleman,J.R. *et al.* (2008) Virus attenuation by genome-scale changes in codon pair bias. *Science*, **320**, 1784–1787.

Condon,A. and Thachuk,C. (2012) Efficient codon optimization with motif engineering. *J. Discrete Algorithms*, **16**, 104–112.

DeNegre,S.T. and Ralphs,T.K. (2009) A branch-and-cut algorithm for integer bilevel linear programs. In: John,W. *et al. Operations Research and Cyber-Infrastructure*. Springer, Berlin Heidelberg, pp. 65–78.

Fox,J.M. and Erill,I. (2010) Relative codon adaptation: a generic codon bias index for prediction of gene expression. *DNA Res.*, **17**, 185–196.

Fuglsang,A. (2003) Codon optimizer: a freeware tool for codon optimization. *Protein Expres. Purif.*, **31**, 247–249.

Gao,W. *et al.* (2004) UpGene: application of a web-based DNA codon optimization algorithm. *Biotechnol. Prog.* **20**, 443-448.

Gaspar,P. *et al.* (2012) EuGene: maximizing synthetic gene design for heterologous expression. *Bioinformatics*, **28**, 2683–2684.

Gaspar,P. *et al.* (2013) mRNA secondary structure optimization using a correlated stem–loop prediction. *Nucleic Acids Res.*, **41**, e73.

Gonzalez-Sanchez,B. *et al.* (2019) Multi-objective artificial bee colony for designing multiple genes encoding the same protein. *Appl.Soft Comput.*, **74**, 90–98.

Gould,N. *et al.* (2014) Computational tools and algorithms for designing customized synthetic genes. *Front. Bioeng. Biotechnol.*, **2**, 41.

Gouy,M. and Gautier,C. (1982) Codon usage in bacteria: correlation with gene expressivity. *Nucleic Acids Res.*, **10**, 7055–7074.

Grote,A. *et al.* (2005) JCat: a novel tool to adapt codon usage of a target gene to its potential expression host. *Nucleic Acids Res.*, **33**, W526–W531.

Guimaraes,J.C. *et al.* (2014) D-tailor: automated analysis and design of DNA sequences. *Bioinformatics*, **30**, 1087–1094.

Gustafsson,C. *et al.* (2004) Codon bias and heterologous protein expression. *Trends Biotechnol.*, **22**, 346–353.

Hoover,D.M. and Lubkowski,J. (2002) DNAWorks: an automated method for designing oligonucleotides for PCR-based gene synthesis. *Nucleic Acids Res.*, **30**, e43.

Jung,S.-K. and McDonald,K. (2011) Visual gene developer: a fully programmable bioinformatics software for synthetic gene optimization. *BMC Bioinformatics*, **12**, 340.

Kudla,G. *et al.* (2009) Coding-sequence determinants of gene expression in *Escherichia coli*. *Science*, **324**, 255–258.
Lithwick,G. and Margalit,H. (2003) Hierarchy of sequence-dependent features associated with prokaryotic translation. *Genome Res.*, **13**, 2665–2673.
Liu,X. *et al.* (2014) COStar: a D-star Lite-based dynamic search algorithm for codon optimization. *J. Theor. Biol.*, **344**, 19–30.
Nakamura,Y. *et al.* (2000) Codon usage tabulated from international DNA sequence databases: status for the year 2000. *Nucleic Acids Res.*, **28**, 292–292.
Papamichail,D. *et al.* (2018) Codon context optimization in synthetic gene design. *IEEE/ACM Trans. Comput. Biol. Bioinform.*, **15**, 452–459.
Poolsap,U. *et al.* (2009) Prediction of RNA secondary structure with pseudoknots using integer programming. *BMC Bioinformatics*, **10**, S38.
Puigbo,P. *et al.* (2007) OPTIMIZER: a web server for optimizing the codon usage of DNA sequences. *Nucleic Acids Res.*, **35**, W126–W131.
Rivas,E. and Eddy,S.R. (1999) A dynamic programming algorithm for RNA structure prediction including pseudoknots. *J. Mol. Biol.*, **285**, 2053–2068.
Sato,K. *et al.* (2011) IPknot: fast and accurate prediction of RNA secondary structures with pseudoknots using integer programming. *Bioinformatics*, **27**, i85–i93.
Satya,R.V. *et al.* (2003) A pattern matching algorithm for codon optimization and CpG motif-engineering in DNA expression vectors. *Proc. IEEE Comput. Soc. Bioinform. Conf.*, **2**, 294–305.
Seffens,W. and Digby,D. (1999) mRNAs have greater negative folding free energies than shuffled or codon choice randomized sequences. *Nucleic Acids Res.*, **27**, 1578–1584.
Sharp,P.M. and Li,W.-H. (1987) The codon adaptation index-a measure of directional synonymous codon usage bias, and its potential applications. *Nucleic Acids Res.*, **15**, 1281–1295.
Skiena,S.S. (2001) Designing better phages. *Bioinformatics*, **17**, S253–S261.
The UniProt Consortium (2018) UniProt: a worldwide hub of protein knowledge. *Nucleic Acids Res.*, **47**, D506–D515.
Villalobos,A. *et al.* (2006) Gene Designer: a synthetic biology tool for constructing artificial DNA segments. *BMC Bioinformatics*, **7**, 285.
Webster,G.R. *et al.* (2017) Synthetic gene design–the rationale for codon optimization and implications for molecular pharming in plants. *Biotechnol. Bioeng.*, **114**, 492–502.
Welch,M. *et al.* (2009) You're one in a googol: optimizing genes for protein expression. *J. R. Soc. Interface*, **6**, S467–S476.
Zuker,M. (2003) Mfold web server for nucleic acid folding and hybridization prediction. *Nucleic Acids Res.*, **31**, 3406–3415.